QNX® Neutrino® RTOS
Utilities Reference

©1996–2014, QNX Software Systems Limited, a subsidiary of BlackBerry. All rights reserved.

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

QNX, QNX CAR, Neutrino, Momentics, Aviage, and Foundry27 are trademarks of BlackBerry Limited that are registered and/or used in certain jurisdictions, and used under license by QNX Software Systems Limited. All other trademarks belong to their respective owners.

Electronic edition published: Thursday, March 6, 2014

Table of Contents

About This Reference

The *Utilities Reference* describes the utilities, manager processes, and configuration files included with the QNX Neutrino RTOS. This reference is intended for everyone from end-users to system administrators.

For information about:	See:
Specific utilities	*A* ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*
An overview of the syntax for utilities	*Utility Conventions* (p. 23)
Environment variables	*Commonly Used Environment Variables*
Targets for GNU utilities	*Selecting the Target System*
A summary of changes to this reference	*What's New in this Reference?*
Terms used in QNX Neutrino documentation	*Glossary*

Your system might not include all of the things that this reference describes, depending on what software you've installed. For example, some utilities are included in a specific Board Support Package (BSP).

For information about licensing, see:

- the QNX Development Suite License Guide at `http://licensing.qnx.com/license-guide/`
- the Third Party License Terms List at `http://licensing.qnx.com/third-party-terms/`
- the matrix of all the versions of all our legal documents at `http://licensing.qnx.com/document-archive/`

Copyright © 2014, QNX Software Systems Limited

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications.

The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL )`
Command options	`-lR`
Commands	`make`
Environment variables	***PATH***
File and pathnames	`/dev/null`
Function names	*exit()*
Keyboard chords	**Ctrl –Alt –Delete**
Keyboard input	`Username`
Keyboard keys	`Enter`
Program output	`login:`
Variable names	*stdin*
Parameters	*parm1*
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective → Show View** .

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Copyright © 2014, QNX Software Systems Limited

Note to Windows users

In our documentation, we use a forward slash (/) as a delimiter in all pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Copyright © 2014, QNX Software Systems Limited

Technical support

Technical assistance is available for all supported products.

To obtain technical support for any QNX product, visit the Support area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Copyright © 2014, QNX Software Systems Limited

Chapter 1
Utility Conventions

Copyright © 2014, QNX Software Systems Limited

Syntax conventions

Most QNX Neutrino utilities follow standard conventions for argument syntax and behavior. These conventions are based on the utility conventions outlined in POSIX 1003.2-1992.

The syntax synopsis for each utility appears at the top of the page of its manual entry. The utility name appears first, followed by other allowed command-line arguments, which include options, option arguments (e.g. "*number*" in -n *number*), and operands (e.g. the names of files to act on).

The syntax synopsis is the only reliable source for information about mutual exclusivity of options and about whether a command-line element is optional or required. This information isn't usually contained in the detailed option listings that appear after the syntax section.

A typical utility syntax line looks like this:

```
utilityname [-abcd] [-o arg | -p arg] infile... outfile
```

The example above shows a utility called `utilityname` that accepts the options -a, -b, -c, and -d — these options may be used alone or in any combination.

The utility also accepts the options -o and -p, both of which require an option argument, and which may not be used together (but may be used with the other options -abcd). The utility requires two or more operands: one or more *infile* and exactly one *outfile*.

Interpreting utility syntax

Here are the main principles at work:

- When utilities have many options, the options may appear grouped together in the syntax like this:

  ```
  utilname [-abcd]
  ```

  which means that the options -a, -b, -c, and -d are supported.
- Options, option arguments, and operands enclosed in brackets ([and]) are optional and can be omitted. Note that the [and] symbols should never be included in the actual command.
- Arguments separated by | are mutually exclusive. Sometimes mutually exclusive arguments that relate to modes of operation are indicated with multiple syntax lines representing the different forms of the command.

Copyright © 2014, QNX Software Systems Limited

- A trailing ellipsis mark (…) after options or operands indicates that the preceding item may be repeated. If the preceding item is optional, the ellipsis indicates that the item may occur zero or more times, e.g.:

  ```
  utility [filename...]
  ```

  If the item is mandatory, the ellipsis indicates it may occur one or more times, e.g.:

  ```
  utility filename...
  ```

Invoking utilities

There are a number of general guidelines to follow when running utilities:

- An option may be followed by another option after a single dash (-) on the command line as long as each preceding option doesn't have an option argument. For example, the option string -abc is equivalent to -a -b -c. However, if -a accepts an option argument, then -abc would be equivalent to -a bc instead.
- Options and their option arguments should be specified with spacing as shown in their documentation. If the documentation says:

  ```
  -n number
  ```

  the *number* should be a separate command-line argument from the -n. But if the documentation refers to:

  ```
  -nnumber
  ```

  then *number* should appear in the same argument as -n without any intervening blanks. Utilities in QNX Neutrino and in POSIX-conforming systems permit both forms in all utilities *unless otherwise stated*, but you'll achieve the greatest portability by using the preferred form. This is particularly important when developing scripts that may be used on multiple (QNX Neutrino and non-QNX Neutrino) platforms.
- Options are usually listed in alphabetical order, but there's no restriction on the order that they may appear in the command line when used, unless otherwise indicated in the documentation for the utility. Note that in some utilities, mutually exclusive options override each other in a “last one wins” manner.

Copyright © 2014, QNX Software Systems Limited

- All options and associated option arguments must precede any operands on the command line. For example, if you want to run the `cp` utility with the -R option, you may enter:

  ```
  cp -R dir1 dir2
  ```

  but not:

  ```
  cp dir1 dir2 -R
  ```

- Decimal integers are accepted when numeric values are required in operands and option arguments, unless otherwise specified. Some utilities may support `0`*octal* and `0x`*hex* numbers as well without being documented as doing so. For this reason, don't precede decimal numbers with leading zeros.
- Integer numerical operands and option arguments must be in the range 0 to 2147483647 unless otherwise specified. If negative numbers are accepted, the acceptable range is -2147483647 to 2147483647.
- The argument -- ("dash dash") may be placed on the command line as a delimiter indicating the end of options and the start of operands. This is particularly useful when the operands themselves might start with a dash. For example, to remove a file named "`-t`", you would use:

  ```
  rm -- -t
  ```

  Utilities that don't accept any options also accept and discard a -- before their operands, unless otherwise indicated.

- Most utilities that accept filenames as operands (and sometimes as option arguments) accept the filename "`-`" to mean standard input, or, when unambiguous from its context, standard output.

Copyright © 2014, QNX Software Systems Limited

File conventions

File pathnames specified on the command line are restricted to 255 characters. Some input files are specifically identified as “text files.” Text files are expected to contain ASCII text in newline-terminated lines that don't exceed 2048 characters, unless otherwise indicated.

Copyright © 2014, QNX Software Systems Limited

Signal conventions

Signal actions are inherited from the process that invokes the utility. Most utilities don't do any special processing upon receipt of a signal, but behave instead according to the system defaults. When a utility performs some action on receipt of a signal other than the default, it's documented as doing so.

Note that temporary files *aren't* left in place after a utility is terminated due to a signal, unless otherwise specified.

Servers and resident processes typically run only as `root` and ignore most signals (such as `SIGPWR`).

Copyright © 2014, QNX Software Systems Limited

Exit status conventions

Utilities normally return zero for successful completion and values greater than zero when unsuccessful. Some utilities return different nonzero numbers according to the reason they failed. Beware of testing for a specific nonzero number to indicate failure. (In most cases utilities that may return different nonzero numbers are explicitly documented as doing so. However, you should not rely on this.)

For some utilities, the exit status may reflect only the success or failure of the last action taken (of many). In these cases, this behavior is explicitly documented in the "Exit status" section.

In the *ksh* (p. 1029), you can use $? to get the exit status of the last command. For more information, see "*Parameters* (p. 1040)" in the documentation for ksh.

Copyright © 2014, QNX Software Systems Limited

Error conventions

Utilities may fail for many reasons ranging from incorrect usage to underlying system failure. The documentation for the utilities doesn't attempt to outline the exact behavior for all possible modes of failure.

In all cases, unless otherwise specified, every error results in a diagnostic message printed to standard error.

When an error occurs, the utility stops the processing of the current operand and proceeds to process the next operand in the sequence. If a utility fails to process one operand but succeeds on others, the exit status still reflects failure. For utilities that recurse through a filesystem (e.g. `find`), if an action cannot be performed on one file within a hierarchy, the utility stops processing that file and goes on to the subsequent files in the hierarchy.

When an unrecoverable error occurs (e.g. insufficient memory), the utility prints a diagnostic message to standard error and exits immediately.

Copyright © 2014, QNX Software Systems Limited

Chapter 2
A

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
awk	*gawk* (p. 869)

This chapter describes the utilities, etc. whose names start with “A”.

Copyright © 2014, QNX Software Systems Limited

ability

Change the ability set of the invoking process (QNX Neutrino)

Syntax:

```
ability [ability-spec...]
```

Runs on:

QNX Neutrino

Options:

ability-spec

Specify an ability to be allowed or disallowed. The *ability-spec* argument is a comma-separated list that contains the following, as required, ignoring the case of the strings:

- the ability identifier, as defined in <sys/procmgr.h>, but omitting the PROCMGR_AID_ prefix (e.g., specify setuid for PROCMGR_AID_SETUID)
- allow or deny
- lock if you want to prevent the process from changing the ability
- root, nonroot, or all to specify the applicable domain
- inherit or noinherit

If the ability accepts a subrange, the above may be followed by a colon and a comma-separated list of subranges, in one of the following forms:

- two numbers separated by a hyphen (e.g., 4-27)
- one number followed by a hyphen (e.g., 4- indicates 4 and greater)
- a single number

Description:

The ability utility lets you allow or deny abilities for the invoking process. You can specify multiple abilities.

If you specify allow, deny, lock, root, nonroot, or all without an ability name, the action applies to all abilities not specifically mentioned in another -A option.

For more information about abilities, see the entry for *procmgr_ability()* in the QNX Neutrino *C Library Reference*.

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

All abilities were successfully parsed and applied.

1

An error occurred.

Examples:

Deny forking while running as `root`, but allow the process to set `_CS_HOSTNAME` when non-`root`:

```
ability root,deny,fork nonroot,allow,confset:2
```

Copyright © 2014, QNX Software Systems Limited

addr2line

Convert addresses into line number/file name pairs (GNU)

Syntax:

addr2line_variant [*options*] [*addr ...*]

where *addr2line_variant* depends on the target platform, as follows:

Target platform	*addr2line_variant*
ARMv7	`ntoarmv7-addr2line`
x86	`ntox86-addr2line`

Runs on:

Linux, Microsoft Windows

Description:

The `addr2line` utility translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

addvariant

Add a new OS, CPU, or VARIANT directory structure to a source tree

Syntax:

```
addvariant [-c] [-i init_lvls] [[os_name] cpu_name] variant_name
```

Runs on:

Linux, Microsoft Windows

Options:

-c

Add the created directory structure to the CVS repository on the next `cvs commit`.

-i *init_lvls*

Create the initial `common.mk` and `Makefile` files in the current working directory. The `Makefile` contains a line defining the level(s) contained in the directory structure. Specify *init_lvls* as `OS`, `OS/CPU`, or `OS/CPU/VARIANT` (use a slash (`/`) or a dash (`-`) to separate multiple levels).

os_name

The name of the operating system to add (e.g. `nto`, `linux`).

cpu_name

The name of the CPU to add (e.g. `arm`, `x86`).

variant_name

The name of the variant to add (e.g. `o`, `a.le`)

Description:

The `addvariant` utility is a shell script that creates a directory structure for your source tree. It also ensures that each level of this structure contains necessary files used by the *make* (p. 1166) utility.

Using `addvariant` to create your variant directory structure enables you to take advantage of the makefile rules of the QNX Neutrino build environment. For more

Copyright © 2014, QNX Software Systems Limited

information on these rules, see the Conventions for Recursive Makefiles and Directories appendix of the QNX Neutrino *Programmer's Guide*.

The project level includes a file called `common.mk`. This file contains any "special" flags and settings needed for compilation and linking.

Each level in the directory structure needs a properly constructed `Makefile` with appropriate macros and include files. At most levels, `Makefile` includes `recurse.mk`, the file used by higher-level makefiles to recurse into lower-levels. The `Makefile` at the lowest level of the directory tree (the variant level) includes the `common.mk` file from the project level instead of `recurse.mk`.

If requested, `addvariant` also adds the directories and files it has created to CVS, ready for your next `cvs commit`.

Dealing with GNU projects

The utility begins by checking to see the projects conform to a GNU-type structure. If the current working directory (CWD) contains files named `configure` and `Makefile.in`, `addvariant` assumes that the project is configured in the GNU style. In that case, it automatically squashes the directory levels (as described below) into a single OS-CPU-VARIANT level and creates `GNUmakefile` files in the newly created directories along with a recursing `Makefile` to take advantage of them.

After you've run `addvariant`, create an executable shell script called `build-hooks` file in the root of the project. This script defines some shell functions that `make` invokes to build your project properly; for more information, see "GNU `configure`" in the Conventions for Recursive Makefiles and Directories appendix of the QNX Neutrino *Programmer's Guide*.

Creating the initial files

The `addvariant` utility either creates and installs standard `Makefile` and `common.mk` files in the CWD, or, if these files already exist, edits them to add the same standard script lines used for recursing.

Creating the subdirectories and files

Starting from the CWD, the `addvariant` utility searches down into the directory tree looking in each `Makefile` for a line starting with `LIST`. This line indicates the particular directory level the `Makefile` is placed in, like this:

- `LIST=OS` (if three levels are specified)
- `LIST=CPU` (if two levels are specified)
- `LIST=VARIANT` (if one level is specified)

The utility then decides whether to create a subdirectory by looking at the:

- LIST macro
- *_name options

Copyright © 2014, QNX Software Systems Limited

- current subdirectories

If needed, `addvariant` then creates an appropriately named subdirectory containing a suitable `Makefile`.

This process continues down into the directory structure until all the required directories have been created and populated with the necessary recursing `Makefile`.

Squashing levels

The `addvariant` utility has the ability to "squash" directory levels together. If you enter the command:

```
addvariant -i OS/CPU/VARIANT nto x86 o
```

`addvariant` creates a recursing `Makefile` in the CWD structure that has a line like this:

```
LIST=OS CPU VARIANT
```

and then creates a single subdirectory called `nto-x86-o`.

Any subsequent invocation of `addvariant` in the tree notices this squashing of the directory levels and automatically generates the appropriate directory structure.

For more information, see Andrew Oram and Steve Talbott, *Managing Projects with make*, O'Reilly and Associates, 1991.

Examples:

Create a two-level directory called `nto-x86/o`:

```
addvariant -i OS/CPU nto x86 o
```

Create the opposite two-level scheme, `nto/x86-o`:

```
addvariant -i OS nto x86/o
```

For detailed examples, see "Examples of creating `Makefile`s" in the Conventions for Recursive Makefiles and Directories appendix of the QNX Neutrino *Programmer's Guide*.

Copyright © 2014, QNX Software Systems Limited

applypatch

Install or uninstall a patch (QNX Neutrino)

Since this utility modifies the installation tree, you must run it as a privileged user. It also requires the environment be properly set. On Linux, if you use sudo (as would be the case on Ubuntu), you must also specify the -E option to preserve the environment. In this case, however, the ***PATH*** environment variable is set to a safe one for security reasons, so you need to specify the full path to the utility:

```
sudo -E $QNX_HOST/usr/bin/applypatch patchfile
```

Syntax:

Install a patch:

```
applypatch [-b] [-c] [-d path] [-F] [-H] [-v] patch_file
```

List the installed patches:

```
applypatch -l
```

Uninstall a patch:

```
applypatch -U num
```

Runs on:

Linux, Microsoft Windows

Options:

-b

Don't make a backup.

If you specify the -b option, you won't be able to uninstall this patch or any patches applied after it. We don't recommend this option for general use.

-c

Extract the patch contents only. No metadata will be recorded, nor will any backup file be generated. This is useful when pulling out individual files for testing, but we don't recommend this option for general use.

-d *path*

Copyright © 2014, QNX Software Systems Limited

Specify the destination path. This path will be the root directory used for extracting the patch contents as well as for storing the patch metadata. The default is the currently active QNX Neutrino installation.

-F

Turn off prompting by forcing a "yes" answer to all queries. Normally newer files aren't overwritten by older files from the patch. This option disables that check. This means locally updated files may be **silently replaced** by an older version from the patch.

-H

(QNX Neutrino 6.5.0 and later) Install all host-side files in the patch. By default, `applypatch` installs only those for the current host OS.

The version of `applypatch` in 6.4.1 installs host-side files for all host OSs.

-I

List the patches, ordered from newest to oldest (based on installation time).

-v

Be verbose. Display some information on progress and activities.

-U ***num***

Uninstall the patch with the specified ID number, *num*.

Due to the nature of the patching process, any patch that was installed after patch ID *num* will be uninstalled as well. In effect, you'll roll back to the state the system was in just before patch ID *num* **and all subsequent patches** were applied.

Description:

The `applypatch` utility installs and uninstalls QNX Neutrino patches, and also lists the installed patches. It's installed in `$QNX_HOST/usr/bin` and supports our current patch *`tar`* (p. 1890) files.

On Windows, run `applypatch` in `cmd.exe`.

This utility first backs up any files which will be overwritten and then extracts the patch files.

Copyright © 2014, QNX Software Systems Limited

If you've applied a sequence of patches, you can uninstall them only in reverse order. If you select a patch (Patch ID *X*) for uninstallation, then any patches installed since Patch ID *X* are also marked for uninstallation. A warning and list of affected patches is printed and confirmation requested for this situation.

Copyright © 2014, QNX Software Systems Limited

aps

Manage adaptive scheduler partitions

Syntax:

```
aps show [-d delay] [-f shorthand] [-l] [-v...]
         [partition_name ...]

aps create -b budget [-B critical_budget] partition_name

aps modify [-b budget] [-B critical_budget] partition_name

aps modify [-y bankruptcy_policy ...] [-S scheduling_policy...]
           [-s security_policy ...] [-w windowsize_ms]
```

Runs on:

QNX Neutrino

Options:

-B *milliseconds*

Specify the critical CPU budget, in milliseconds. The default is 0.

-b *budget*

Specify the CPU budget as a percentage.

-d *delay*

The delay period, in tenths of a second, when using the -l option. The default is 50.

-f *shorthand*

Display the information specified by *shorthand*:

- `all` — all the below
- `overall_stats` — information about the last bankruptcy
- `scheduler` — parameters for the thread scheduler, including the current security setting, bankruptcy policy, and the size of the averaging window
- `partitions` — information about the partitions, including their names, IDs, parent IDs, budgets, critical budgets, and the process and thread IDs of the last thread to go bankrupt

Copyright © 2014, QNX Software Systems Limited

- `usage` — the amount of budget and critical budget that each partition is currently using

The default is `usage`.

-l

("el") Loop mode; display the information at the interval specified by the -d option.

-S *scheduling_policy* ...

Specify the policies for the adaptive partitioning scheduler. Each *scheduling_policy* must be one of:

- `normal`
- `freetime_by_ratio`

The default is `normal`. For more information about the policies, see "Scheduling policies" in the entry for *SchedCtl()* in the QNX Neutrino *C Library Reference*.

-s *security_policy* ...

Specify the security policies to add to the system. Each *security_policy* must be one of:

- `root0_overall`
- `root_makes_partitions`
- `sys_makes_partitions`
- `parent_modifies`
- `nonzero_budgets`
- `root_makes_critical`
- `sys_makes_critical`
- `root_joins`
- `sys_joins`
- `parent_joins`
- `join_self_only`
- `partitions_locked`
- `recommended`
- `flexible`
- `basic`
- `none`

Copyright © 2014, QNX Software Systems Limited

The default is `none`. For more information about the policies, see the description of `SCHED_APS_ADD_SECURITY` in the entry for *SchedCtl()* in the QNX Neutrino *C Library Reference*.

Once you've added a security policy, you can't remove it, except by rebooting the system.

-v...

Be verbose; display more information with the show command:

- -v — display the budget usage over the last averaging window, window 2 (typically 10 times the length of the averaging window), and window 3 (typically 100 times the length of the averaging window)
- -vv — display the budget usage and critical budget usage over the last averaging window, window 2, and window 3

-w *windowsize_ms*

Set the size of the averaging window, in milliseconds, for the system. You can set the window size to any value from 8 ms to 400 ms.

If you change the tick size of the system at runtime, do so before defining the adaptive partitioning scheduler's window size. That's because QNX Neutrino converts the window size from milliseconds to clock ticks for internal use.

For more information, see "Choosing the window size" in the System Considerations chapter of the Adaptive Partitioning *User's Guide*.

-y *bankruptcy_policy* ...

Set the bankruptcy policy for the system to the specified items. Each *bankruptcy_policy* must be one of:

- `cancel_budget` — set the offending partition's critical budget to zero, which forces the partition to be scheduled by its percentage CPU budget only. This also means that a second bankruptcy can't occur.
- `log` — not currently implemented.
- `reboot` — cause the system to crash with a brief message identifying the offending partition. This is the most severe response, suggested for use while testing a product, to make sure bankruptcies are never ignored. You probably shouldn't use this option in your finished product.
- `basic` — deliver bankruptcy-notification events and make the partition out-of-budget for the rest of the scheduling window (nominally 100 ms).

Copyright © 2014, QNX Software Systems Limited

- `recommended` — the combination of `cancel_budget` and `log`.
- `none` — do nothing.

The default is `basic`. For more information about the policies, see "Handling bankruptcy" in the entry for *SchedCtl()* in the QNX Neutrino *C Library Reference*.

Description:

Use the `aps` command to create, modify, and query adaptive partitions from the command line, as well as to set the averaging window, and the security and bankruptcy policies for the entire system.

You can't include slashes (`/`) in a partition name.

To launch an application into a partition, use the -Xaps option to the *on* (p. 1417) command.

Examples:

Create a partition called `Drivers` with a budget of 20% and a critical budget of 5 milliseconds:

```
aps create -b 20 -B 5 Drivers
```

Change the `Drivers` partition's budget to 25% and its critical budget to 7 milliseconds:

```
aps modify -b 25 -B 7 Drivers
```

Specify a bankruptcy policy of `recommended` and a security policy of `root_makes_partitions` for the entire system:

```
aps modify -y recommended -s root_makes_partitions
```

Display the amount of the budget and critical budget that the partitions are using, every 2 seconds:

```
aps show -l -d 20 -f usage
```

Since `usage` is the default shorthand for the -f option, the above command is the same as:

```
aps show -l -d 20
```


Copyright © 2014, QNX Software Systems Limited

ar

Create and maintain library archives (POSIX)

Syntax:

```
ar_variant -key_letter[mod [relpos]] archive [member…]
ar_variant -M [ <mri-script ]
```

where *ar_variant* depends on the target platform, as follows:

Target platform	*ar_variant*
ARMv7	`ntoarmv7-ar`
x86	`ntox86-ar`

Runs on:

Linux, Microsoft Windows

Description:

The `ar` program creates and modifies archives, and extracts members from them. An *archive* is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called *members* of the archive).

The original files' contents, mode (permissions), timestamp, owner, and group are preserved in the archive; you can restore them on extraction. The `ar` utility can maintain archives whose members have names of any length.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

arp

Address Resolution Protocol (ARP) display and control

Syntax:

```
arp [-n] hostname
arp [-nv] -a
arp [-v] -d -a
arp [-v] -d hostname [pro [iface_name]]
arp -s hostname ether_addr [temp] [pub [pro [iface_name]]]
arp -f filename
```

Runs on:

QNX Neutrino

Options:

-a

Display all of the current ARP entries.

-d *hostname*

Delete an entry for the specified host. Only the superuser can use this option.

-f *filename*

Load the ARP cache with entries found in the specified file. Entries in the file should be of the form:

```
hostname ether_addr [pub] [temp]
```

with argument meanings as given for the -s option.

-n

Don't try to resolve hostnames.

-s *hostname ether_addr* [temp] [pub [pro [*iface_name*]]]

Create an ARP entry for the host *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six hex bytes separated by colons.

If pub is given, the entry is “published.” That is, this system acts as an ARP server, responding to requests for *hostname*, even though the IP address that's mapped to *hostname* isn't the address of this system.

Copyright © 2014, QNX Software Systems Limited

The entry is permanent unless the word `temp` is given in the command.

Specify the `pro` string to create or delete a proxy-only ARP entry.

The *iface_name* argument is the name of the interface (e.g. `fxp0`).

-v

Be verbose.

hostname

A host name or the IP address.

Description:

The `arp` utility displays and modifies the Internet-to-Ethernet address translation tables used by the Address Resolution Protocol (ARP).

With no options, the utility displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

License:

This utility is based on copyright software of The Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

asa

Translate line-printer control sequences to newlines/form feeds (POSIX)

Syntax:

```
asa [filename...]
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `asa` utility writes its input files to standard output, mapping carriage control characters from the files to line-printer control sequences. If you specify `-` for a file name, `asa` reads from standard input.

All characters in column position 1 are removed from the input, and the following actions are performed, depending on the character:

Space

The rest of the line is output without change.

0

A newline (`\n`) character is output, followed by the rest of the input line.

1

A form-feed character (`0x0C`) is output, followed by the rest of the input line.

+

The newline of the previous line is replaced with a carriage return (`0x0D`), followed by the rest of the input line.

Any other character in column 1 is left unchanged.

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

All input files were output successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

/etc/autoconnect

Automatic TCP/IP connection-configuration script

Name:

`/etc/autoconnect`

Description:

The `/etc/autoconnect` script is run when an application needs to establish a TCP/IP connection to a remote host. This file can be in any form (e.g. a shell script or an executable), and contains all of the necessary commands required to create the connection.

To activate the script, define the environment variable ***AUTOCONNECT*** and set its value to 1.

If there's no route to a remote host (see *route* (p. 1680), *ifconfig* (p. 957), or the options to *io-pkt** (p. 1007)), or there are no nameservers defined (see */etc/nsswitch.conf* (p. 1369)) and a hostname can't be resolved, the autoconnect script is run. The exit status of the script determines whether or not a retry is attempted:

Zero

The socket library attempts the action again.

Nonzero

It doesn't retry because the script failed.

One time you might use this feature is when you have a dialup ISP account for internet access. The ppp link is established only when an application needs to reach a host over the link. When the link is terminated depends on inactivity timeouts specified by the client or server, errors, or other events. The autoconnect script only launches commands to establish a connection; it doesn't terminate the connection.

Suppose that a host is configured with only the localhost interface, and no nameservers. You need to create a script:

```
pppd connect "/bin/chat -v -f /etc/chat" defaultroute \
+resconf 115200 updetach /dev/ser2

exit
```

The *chat* (p. 99) utility is used to dial the service provider. The important option here is the updetach option. This option daemonizes *pppd* (p. 1560) after the PPP interface has been configured. This way, the script doesn't exit until the interface is configured. If an application attempts to resolve a hostname, the application blocks while a connection to an ISP is established, which provides a nameserver and a default route.

Copyright © 2014, QNX Software Systems Limited

When the script exits with a status of zero, the socket library retries and the application continues, assuming that the function succeeded. If an exit status of non-zero is returned, the socket library returns the original error to the application.

Copyright © 2014, QNX Software Systems Limited

Chapter 3
B

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “B”.

Copyright © 2014, QNX Software Systems Limited

basename

Return the nondirectory portion of pathname (POSIX)

Syntax:

```
basename string [suffix]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

string

A string of text.

suffix

A string of text.

Description:

The basename utility is useful primarily for extracting the filename portion of a pathname, but since it performs only string operations, you can use it with any string.

The basename utility prints to standard output a substring of its *string* argument, plus a newline character. The basename utility forms this substring by doing the following, in order:

1. Discarding any trailing slash (/) characters.
2. Discarding all characters up to and including the last slash character.
3. Deleting the *string* argument's suffix, provided you've specified a *suffix* operand that's identical to the *string* operand's suffix.

   If *suffix* is equal to the remaining string, the suffix isn't removed (e.g. if *suffix* is `prog.c` and the remaining string is `prog.c`, the suffix `.c` isn't removed).

The result is a null string only if *string* is a null string (""). In this case, basename outputs a single newline character.

If *string* consists entirely of slash characters, basename prints a single slash, followed by a newline character.

You'll use the basename utility most often within shell scripts, where it's normally invoked inside back-ticks (`` `...` ``), or contained in `$(...)`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Command:	Output:
`basename .`	`.`
`basename /usr/src/prog.c`	`prog.c`
`basename /usr/src/prog.c .c`	`prog`
`basename /usr/src/prog.c .a`	`prog.c`
`basename /usr/src/`	`src`
`basename ...//[fred]`	`[fred]`

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

bc

"Bench calculator" arbitrary-precision arithmetic language (POSIX)

Syntax:

```
bc [-l] [file...]
```

Runs on:

QNX Neutrino

Options:

-l

("el") Include a library of defined math functions and set the scale to 20. (See *"Library functions,"* (p. 62) below.)

file

The pathname of a text file containing commands and function definitions to be interpreted by bc. If any files are specified, bc processes those files, then reads from the standard input.

Description:

The bc utility is an interactive, programmable calculator that supports a complete set of control structures, including functions.

The bc utility supports 26 functions, 26 simple variables, and 26 array variables. Each array may have up to 2048 elements. In addition, the utility performs arithmetic operations using a radix 100 number system with user-definable precision. When you specify the precision, the numbers are *exact* to that precision, unlike binary floating-point representation where rounding errors may compromise accuracy. The utility also operates in different bases, so you can easily convert numbers from one base to another.

Many common programming language constructs are supported, including:

- `if`, `while`, and `for` statements
- user-defined functions with parameters
- local variables.

The bc utility provides no support for character or string manipulation. The syntax of bc is derived from C, but doesn't constitute a programming language.

Copyright © 2014, QNX Software Systems Limited

Let's look at a very simple bc program:

```
"hello, world\n"
```

When this program is run, the statement "hello, world" is echoed with a newline character. In general, the result of any expression that isn't assigned to a variable or used in a control structure is echoed. For example, the following statement:

```
5^2
```

causes bc to respond with 25.

Note that the caret (^) is an integer exponentiation operator: *a*^*b* is *a* raised to the *b*th power, where *b* is truncated to an integer in the range -2^31 to +2^31. All of the "usual" arithmetic operators (+, -,*,/,%) are available, but % is the remainder, not the modulus.

Bases

Two special builtin variables let you choose the base in which numbers are input and output:

ibase

Input base.

obase

Output base.

When numbers are recognized, the input base determines the significance of each digit. For example, the value 66 in base 10 is:

```
    6 × 10^0 =  6
+   6 × 10^1 = 60
               --
               66
```

whereas in base 8, it's:

```
    6 × 8^0  =  6
+   6 × 8^1  = 48
               --
               54
```

Note that hex numbers are recognized in the different bases and that their values also change with the base. For example, the number FF is valid in base 10 and has the decimal value 165:

```
    F(=15) × 10^0 =  15
+   F(=15) × 10^1 = 150
                    ---
                    165
```

Setting the output base results in one of two numerical formats:

Copyright © 2014, QNX Software Systems Limited

- For bases in the range 2 to 16, the format consists of the alphabet 0-9,A-F.
- For bases in the range 17 to 10000, the format consists of a series of decimal digits with groups separated by spaces. In this format, each group represents a digit of significance, thus the number 61 in `obase=20` is printed as:

  ```
  3 1
  ```

  which should be interpreted as:

  ```
      1 × 20^0 =  1
  +   3 × 20^1 = 60
                 --
                 61
  ```

Variables

The `bc` utility supports 26 simple variables, named *a* through *z* (case is significant), and 26 arrays, also named *a* through *z*. The context determines whether the simple variable or the array is selected. All variables are auto-initialized to 0 and are always available (i.e. there's no declaration statement).

Functions may create local variables (with the `auto` statement) that hide the global variables of the same name while the function is executing. Arrays are limited to 2048 elements. Variables may be referenced (right-valued) or assigned to (left-valued).

Assignment operators

The assignment operator has many variations, as in the C language. For example:

```
a += x
```

means the same as:

```
a = a + x
```

All of the binary operators have a corresponding assignment operator.

An assignment expression (e.g. `a=10`) has a *right value* that's the new value of *a*. Thus:

```
a = b = 10
```

assigns 10 to b, then assigns b to a.

The increment and decrement operators may be applied only to variables. The statement:

```
b = a++
```

is semantically the same as:

```
b = a; a = a + 1;
```

and the statement:

```
b = ++a
```


Copyright © 2014, QNX Software Systems Limited

is semantically the same as:

```
a = a + 1; b = a;
```

The following table summarizes operator precedence and associativity:

Operator	Associativity	Class
++, --	Nonassociative	(increment, decrement)
-	Right to left	(unary minus)
^	Right to left	(exponentiation)
*, /, %	Left to right	(multiplicative)
+, -	Left to right	(additive)
==, <=, >=, !=, <, >	Left to right	(comparative)
= +=, -=, *=, /=, %=, ^=,	Right to left	(assignment)

The `if` statement

The `if` statement takes the following form:

```
if (expr) statement
```

The *statement* is executed only if *expr* evaluates to a nonzero quantity. Regardless of the value of *expr*, the next statement is executed.

```
if (i > 2047) {
    "i exceeds array limit"
    i = 2047;         /* limit index */
}
a[i]      /* always print value */
```

Iteration statements

The `bc` utility supports two iteration statements: `while` and `for`. In both of these structures, `break` means “exit the loop immediately,” and `continue` means “pretend you have finished executing *statement*.”

The `while` statement has the general form:

```
while (expr) statement
```

and is executed following this algorithm:

1. If *expr* is false, go to 4.
2. Execute *statement*.
3. Go to 1.
4. Proceed.

Copyright © 2014, QNX Software Systems Limited

In the `while` statement, `continue` and `break` are analogous to "Go to 1" and "Go to 4," respectively.

The `for` statement has the general form:

```
for (expr1;expr2;expr3) statement
```

and is executed following this algorithm:

1. *expr1*
2. If *expr2* is false, go to 6.
3. Execute *statement*.
4. Execute *expr3*.
5. Go to 2.
6. Proceed.

In the `for` statement, `continue` and `break` are analogous to "Go to 4" and "Go to 6," respectively.

User-defined functions

In `bc`, you can define named functions that support parameters, local variables, and recursion. A function name is a single lowercase character (a to z). A function has the general form:

```
define name(parameter-list) { function-body }
```

You can't remove or replace functions; it's an error to attempt to do so. Functions can call other functions, including themselves.

Functions can accept an arbitrary number of parameters, which may be either simple or array variables.

Parameters are passed by value, that is, the function is given a private copy of all parameters that remains in effect until the function returns. Notice that arrays are also passed by value; this differs from the C calling convention.

The first statement in a function can be an `auto` statement, which creates "local" variables for the function. The auto statement has this general form:

`auto` [*variable*|*array*]...

The local variables are used in place of the global variables of the same name until the function returns; the local variables are also initialized to zero (see the example below).

Functions can contain a `return` statement. The `return` statement causes the function to exit, discarding any local variables and parameters. The statement may be in either of two forms:

`return`

Copyright © 2014, QNX Software Systems Limited

Or

`return` [*retval*]

where *retval* is a constant or a variable name. If no specific `return` statement is given, or if *retval* is omitted, a value of 0 is returned.

The following example shows various constructs being used in functions:

```
/* function s(b[], n)
 * return the sum of the first n elements of b[]
 */
define s(b[],n) {
auto t,i;     /* note that the ';'s are for style,
                 and aren't required. */

    if (n > 2048) {
        "Array length out of bounds...\n"
        return 0;
    }
    for (i=0; i < n; i++) {
        t += b[i];
    }
    return t;
}
```

The function *s()* is introduced by the `define` statement, which also reveals the parameter types. It's important that all functions be called with the appropriate types and number of parameters. Failure to do so results in unpredictable errors.

All parameters in bc are passed by value, including arrays. This is unlike C, where arrays are always passed by reference.

The `auto` statement introduces the locally scoped variables *t* and *i*. You should use automatic variables to prevent unwanted side effects of using global variables.

Builtin variables and functions

Several builtin control variables affect how bc operates. The following variables may be set or queried:

Name	Function	Range	Default
scale	Minimum precision maintained in calculations	0..100	0
ibase	Radix for input	2..10000 numbers	10
obase	Radix for output	2..10000 numbers	10

The bc utility provides the following builtin functions:

Copyright © 2014, QNX Software Systems Limited

Function	Returns
sqrt(*expression*)	Square root of the value
length(*expression*)	Number of significant decimal digits in the expression
scale(*expression*)	Number of decimal digits to the right of the decimal marker

Library functions

The bc library is enabled if you invoke bc with the -l option, or if you include the interpretation file `/usr/lib/bclib.b` in the command line. The library sets the scale to 20, which you can override by assigning a new value. The library defines the following functions:

Syntax	Function
a(*expression*)	Arctangent (in radians)
c(*expression*)	Cosine (in radians)
e(*expression*)	Exponential function
k(*expr1*, *expr2*)	Bessel function of integer order
l(*expression*)	Natural logarithm
s(*expression*)	Sine (in radians)

See also:

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978

Files:

`/usr/lib/bclib.b`

Contains the bc library, a set of function definitions that are loaded automatically when the -l option is specified. This file must be present for the -l option to work.

Exit status:

0

Successful completion.

> 0

Copyright © 2014, QNX Software Systems Limited

An error occurred.

Caveats:

The `bc` utility's syntax contains a few ambiguities:

- The notion of a *statement* in `bc` is either a structured statement or an expression followed by a newline or a semicolon (`;`). An expression may be nil, thus a single newline forms a statement. This can lead to unexpected behavior, as in the following example, where the `while` statement never terminates:

  ```
  a=0
  while (a < 100)
      a++
  a
  ```

  Although the intent is obvious, the newline at the end of the `while` statement comprises the body of the loop, thus *a* is never incremented. The C equivalent of this example is:

  ```
  a=0;
  while (a < 100);
  ```

  The following are “correct” versions:

  ```
  a=0
  while (a < 100) a++
  ```

  Or

  ```
  while (++a < 100)
  ```

  Or

  ```
  while (a < 100) {
      a++
  }
  ```

- The `quit` statement causes `bc` to exit when it encounters the statement, not when it attempts to execute it. Thus, the following causes `bc` to exit immediately:

  ```
  if (0) {
      quit
  }
  ```

  and is equivalent to:

  ```
  quit
  ```

Copyright © 2014, QNX Software Systems Limited

bison

General-purpose parser generator (GNU)

Syntax:

`bison [`*`options`*`] `*`input_file`*

Runs on:

Linux, Microsoft Windows

Options:

For a complete listing, see the documentation at *http://www.gnu.org/software/bison/manual/*.

Description:

The `bison` utility is a general-purpose parser generator that converts an annotated context-free grammar into a parser for that grammar. For detailed documentation, see *http://www.gnu.org/software/bison/manual/*.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

bootpd

Internet boot protocol server

You must be `root` to start this server.

Syntax:

```
bootpd [-dpsT] [-t timeout] [configfile]
```

Runs on:

QNX Neutrino

Options:

-d

Increase the level of debugging output (-dd to be more verbose).

-s

Run in standalone configuration. Use this option if `bootpd` is not being started by `inetd` (e.g. when started in a `sysinit` file).

-T

Do not remove unknown vendor-specific information. The default is to remove the vendor-specific section of the response packet if unknown, or remove unknown options if the vendor is known.

-t *timeout*

When not running standalone, this option specifies the time in minutes that `bootpd` will wait for the next boot request before exiting to conserve system resources. When the next boot request arrives, `inetd` will restart `bootpd`. If you don't want `bootpd` to exit, give *timeout* a value of 0. The default timeout is 15 minutes.

configfile

Specify a configuration file (the default file is `/etc/bootptab`).

Copyright © 2014, QNX Software Systems Limited

Description:

The `bootpd` server implements an Internet Boot Protocol server as defined in *RFC 951* and *RFC 1048*.

It's normally invoked by the *`inetd`* (p. 977) daemon via the following line in the *`/etc/inetd.conf`* (p. 979) file:

```
bootps  dgram  udp wait root /usr/sbin/bootpd  bootpd
```

Note that the descriptions in the default `inetd.conf` file are commented out; uncomment the ones that you want to use.

This method conserves system resources: `bootpd` is started only when a boot request arrives, and if it doesn't receive another boot request within fifteen minutes (default) of the last one received, it exits. You can use the -t option to specify a different timeout value in minutes (e.g. `-t 20`). A timeout value of zero means forever.

Rather than wait for a boot request, `bootpd` can be started independently of `inetd`. This is probably the desired mode of operation for large network installations with many hosts.

When using the -s option, `bootpd` and `inetd` may compete for the same port. Make sure to comment out the `bootps` entry in the `/etc/inetd.conf` file. In this case, the -t option has no effect, because `bootpd` never exits.

Upon startup, `bootpd` first reads its configuration file, *`/etc/bootptab`* (p. 68), and then begins listening for BOOTREQUEST packets.

The server rereads its configuration file when it receives a hangup signal (`SIGHUP`) or when it receives a `bootp` request packet and detects that the file has been updated. Hosts may be added, deleted, or modified when the configuration file is reread.

Based on:

RFC 951, *RFC 1048*, *RFC 1084*, *Assigned Numbers*

Files:

`/etc/bootptab`

Boot protocol server configuration file.

`/etc/services`

Service name database.

The `bootpd` daemon requires the `libsocket.so` shared library.

Errors:

Reported to system log.

Copyright © 2014, QNX Software Systems Limited

License:

This utility is based on software of Carnegie Mellon University; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Caveats:

Individual host entries must not exceed 1024 characters.

Copyright © 2014, QNX Software Systems Limited

/etc/bootptab

Boot protocol server configuration file

Name:

/etc/bootptab

Description:

The /etc/bootptab file is used by the *bootpd* (p. 65) daemon.

The configuration file has a format similar to that of termcap, in which two-character case-sensitive tag symbols are used to represent host parameters. These parameter declarations are separated by colons (:). The general format is:

hostname:*tg*=*value*… :*tg*=*value*… :*tg*=*value*…

where *hostname* is the actual name of a *bootp* client and *tg* is a two-character tag symbol. Most tags must be followed by an equals sign (=) and a value as above. Some may also appear in a Boolean form with no value (e.g. :tg:). The currently recognized tags are:

bf

Bootfile.

bs

Bootfile size. This can be:

- a decimal, octal, or hexadecimal integer (specifying the size of the bootfile in 512-octet blocks)

  Or

- the keyword auto, which causes the server to automatically calculate the bootfile size at each request.

If bs is specified as a boolean, auto is assumed.

cs

Cookie server address list (a whitespace-separated list of IP addresses).

ds

Domain name server address list (a whitespace-separated list of IP addresses).

gw

Copyright © 2014, QNX Software Systems Limited

Gateway address list (a whitespace-separated list of IP addresses).

ha

Host hardware address. This *must* be specified in hexadecimal (optional periods and/or a leading `0x` may be included for readability). The `ha` tag must follow the `ht` tag (either explicitly or implicitly; see `tc` below).

hd

Bootfile home directory.

hn

Send hostname. This is strictly a boolean tag; it doesn't take the usual equals sign and value. Its presence indicates that the hostname should be sent to *RFC 1048* clients. The `bootpd` daemon attempts to send the entire hostname as it's specified in the configuration file. If the name doesn't fit into the reply packet, it's shortened to just the *host* field (up to the first period, if present) and then tried. The server never sends an arbitrarily truncated hostname (if nothing reasonable fits, nothing is sent).

ht

Host hardware type (see Assigned Numbers RFC). This tag specifies the hardware type code as either an unsigned decimal, octal, or hexadecimal integer or as one of the following symbolic names:

- `ethernet` or `ether` for 10M Ethernet
- `ieee802`, `tr`, or `token-ring` for IEEE 802 networks.

im

Impress server address list (a whitespace-separated list of IP addresses).

ip

Host IP address (a single IP address).

lg

Log server address list (a whitespace-separated list of IP addresses).

lp

LPR server address list (a whitespace-separated list of IP addresses).

ns

IEN-116 name server address list (a whitespace-separated list of IP addresses).

Copyright © 2014, QNX Software Systems Limited

rl

Resource location protocol server address list (a whitespace-separated list of IP addresses).

sa

TFTP server address the client should use (useful with multi-homed servers). This tag takes a whitespace-separated list of IP addresses.

sm

Host subnet mask (a single IP address).

tc

Table continuation (points to similar “template” host entry).

td

TFTP root directory used by secured TFTP server.

to

Time offset. This can be:

- a signed decimal integer specifying the client's timezone offset (in seconds from UTC)

  Or

- the keyword `auto`, which uses the server's timezone offset.

If `to` is specified as a boolean, `auto` is assumed.

ts

Time server address list (a whitespace-separated list of IP addresses).

vm

Vendor magic cookie selector. This can be one of the following keywords:

- `auto` (indicating that vendor information is determined by the client's request)
- `rfc1048` (which always forces an *RFC 1048*-style reply)
- `cmu` (which always forces a CMU-style reply).

There's also a generic tag, `T`*n*, where *n* is an *RFC 1048* vendor field tag number. This lets you take advantage of future extensions to *RFC 1048* without being forced to modify `bootpd` first. Generic data may be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII characters. The length of the

Copyright © 2014, QNX Software Systems Limited

generic data is automatically determined and inserted into the proper field(s) of the *RFC 1048*-style `bootp` reply.

All IP addresses are specified in standard Internet "dot" notation and may use decimal, octal, or hexadecimal numbers (octal numbers begin with `0`, hexadecimal numbers begin with `0x` or `0X`).

The hostname, home directory, and bootfile are ASCII strings that may be optionally surrounded by double quotes (`"`). The client's request and the values of the `hd` and `bf` symbols determine how the server fills in the *bootfile* field of the `bootp` reply packet.

If the client specifies an absolute pathname and that file exists on the server machine, then that pathname is returned in the reply packet. If the file can't be found, the request is discarded and no reply is sent. If the client specifies a relative pathname, a full pathname is formed by prepending the value of the `hd` tag and testing for existence of the file. If the `hd` tag isn't supplied in the configuration file or if the resulting boot file can't be found, then the request is discarded.

Clients that specify null boot files always elicit a reply from the server. The exact reply again depends on the `hd` and `bf` tags. If the `bf` tag gives an absolute pathname and the file exists, then that pathname is returned in the reply packet. If the `hd` and `bf` tags together specify an accessible file, then that filename is returned in the reply. If a complete filename can't be determined or if the file doesn't exist, then the reply contains a zeroed-out *bootfile* field.

In all these cases, the file must have its public read access bit set, since this is required by *`tftpd`* (p. 1958).

Many host entries often share common values for certain tags (such as name servers, etc.). Rather than repeatedly specify these tags, you can use the `tc` (table continuation) mechanism to list a full specification for one host entry that can be shared by others. The template entry is often a dummy host that doesn't actually exist and never sends `bootp` requests. This feature is similar to the `tc` feature of `termcap` for similar terminals.

Note that `bootpd` allows the `tc` tag symbol to appear anywhere in the host entry, unlike `termcap`, which requires it to be the last tag. Information explicitly specified for a host always overrides information implied by a `tc` tag symbol, regardless of its location within the entry. The value of the `tc` tag may be the hostname or IP address of any host entry previously listed in the configuration file.

Sometimes it's necessary to delete a specific tag after it's been inferred via `tc`. You can do this by using the construction tag `@`, which removes the effect of the tag as in `termcap`.

Copyright © 2014, QNX Software Systems Limited

For example, to completely undo an IEN-116 name server specification, put the following at an appropriate place in the configuration entry:

```
:ns@:
```

After removal with @, a tag is eligible to be set again through the `tc` mechanism.

Blank lines and lines beginning with a pound sign (**#**) are ignored in the configuration file. Host entries are separated from one another by newlines; a single host entry may be extended over multiple lines if the lines end with a backslash (\). Lines can be longer than 80 characters.

Tags may appear in any order, with the following exceptions:

- the hostname must be the very first field in an entry
- the hardware type must precede the hardware address.

Here's a sample `/etc/bootptab` file:

```
# Sample bootptab file

default1:\
    :hd=/usr/boot:bf=null:\
    :ds=128.2.35.50 128.2.13.21:\
    :ns=0x80020b4d 0x80020ffd:\
    :ts=0x80020b4d 0x80020ffd:\
    :sm=255.255.0.0:gw=0x8002fe24:\
    :hn:vm=auto:to=-18000:\
    :T37=0x12345927AD3BCF:T99="Special ASCII string":

carnegie:ht=6:ha=7FF8100000AF:ip=128.2.11.1:tc=default1:
baldwin:ht=1:ha=0800200159C3:ip=128.2.11.10:tc=default1:
wylie:ht=1:ha=00DD00CADF00:ip=128.2.11.100:tc=default1:
arnold:ht=1:ha=0800200102AD:ip=128.2.11.102:tc=default1:
bairdford:ht=1:ha=08002B02A2F9:ip=128.2.11.103:tc=default1:
bakerstown:ht=1:ha=08002B0287C8:ip=128.2.11.104:tc=default1:

# Special domain name server for next host
butlerjct:ht=1:ha=08002001560D:ip=128.2.11.108:ds=128.2.13.42:tc=default1:

gastonville:ht=6:ha=7FFF81000A47:ip=128.2.11.115:tc=default1:
hahntown:ht=6:ha=7FFF81000434:ip=128.2.11.117:tc=default1:
hickman:ht=6:ha=7FFF810001BA:ip=128.2.11.118:tc=default1:
lowber:ht=1:ha=00DD00CAF000:ip=128.2.11.121:tc=default1:
mtoliver:ht=1:ha=00DD00FE1600:ip=128.2.11.122:tc=default1:
```

The `bootpd` daemon looks in `/etc/services` to find the port numbers it should use. Two entries are extracted:

- `bootps` — the bootp server listening port
- `bootpc` — the destination port used to reply to clients.

If the port numbers can't be determined this way, they're assumed to be 67 for the server and 68 for the client.

Copyright © 2014, QNX Software Systems Limited

Caveats:

As a QNX Neutrino extension, the argument to the `bf` tag can start with an "or bar" character (`|`). If it does, then the following is taken to be a command to spawn in order to get a boot image:

```
bf=|cd /boot; mkifs build/node1:
```

If you use this extension, *`tftpd`* (p. 1958) must not be started as `root`. One choice is to create the user `tftp` and start `tftpd` as that user. You could also create and use the user *`ftp`* (p. 848), but that opens up your machine to anonymous `ftp`. For information on how to change the user as which `tftpd` starts, see */etc/inetd.conf* (p. 979).

Copyright © 2014, QNX Software Systems Limited

brconfig

Configure network bridge parameters

Syntax:

```
brconfig -a
```

or:

```
brconfig  bridge [command [args ...]]
```

Runs on:

QNX Neutrino

Options:

-a

Display the status of all bridge devices present on the system. This flag is mutually exclusive with all other subcommands.

All other operations require that a bridge be specified. If a bridge is specified with no subcommands, the status of that bridge is displayed. The following subcommands are available:

up

Start forwarding packets on the bridge.

down

Stop forwarding packets on the bridge.

add ***interface***

Add the *interface* named by interface as a member of the bridge. The interface is put into promiscuous mode so that it can receive every packet sent on the network.

delete ***interface***

Remove the interface named by interface from the bridge. Promiscuous mode is disabled on the interface when it's removed from the bridge.

maxaddr ***size***

Set the size of the bridge address cache to size. The default is 100 entries.

Copyright © 2014, QNX Software Systems Limited

timeout *seconds*

Set the timeout of address cache entries to the given number of seconds. If *seconds* is zero, then address cache entries don't expire. The default is 1200 seconds.

deladdr *address*

Delete *address* from the address cache.

flush

Delete all dynamically-learned addresses from the address cache.

flushall

Delete all addresses, including static addresses, from the address cache.

discover *interface*

Mark an interface as a "discovering" interface. When the bridge has no address cache entry (either dynamic or static) for the destination address of a packet, the bridge forwards the packet to all member interfaces marked as "discovering." This is the default for all interfaces added to a bridge.

-discover *interface*

Clear the "discovering" attribute on a member interface. For packets without the "discovering" attribute, the only packets forwarded on the interface are broadcast or multicast packets and packets for which the destination address is known to be on the interface's segment.

learn *interface*

Mark an interface as a "learning" interface. When a packet arrives on such an interface, the source address of the packet is entered into the address cache as being a destination address on the interface's segment. This is the default for all interfaces added to a bridge.

-learn *interface*

Clear the "learning" attribute on a member interface.

stp *interface*

Enable the IEEE 802.1D Spanning Tree Protocol (STP) on the interface. Spanning Tree is used to detect and remove loops in a network topology.

-stp *interface*

Copyright © 2014, QNX Software Systems Limited

Disable the Spanning Tree Protocol on the interface. This is the default for all interfaces added to a bridge.

`maxage` ***seconds***

Set the time that a Spanning Tree Protocol configuration is valid. The default is 20 seconds, the minimum 1 second, and the maximum 255 seconds.

`fwddelay` ***seconds***

Set the time that must pass before an interface begins forwarding packets when Spanning Tree is enabled. The default is 15 seconds, the minimum 1 second, and the maximum 255 seconds.

`hellotime` ***seconds***

Set the time between broadcasting of Spanning Tree protocol configuration messages. The default is 2 seconds, the minimum 1 second, and the maximum 255 seconds.

`priority` ***value***

Set the bridge priority for Spanning Tree. The default is 32768. Allowed numerical values range from 0 (highest priority) to 65535 (lowest priority).

`ifpriority` ***interface value***

Set the Spanning Tree priority of the *interface* to *value*. The default is 128, the minimum 0, and the maximum 255.

`ifpathcost` ***interface value***

Set the Spanning Tree path cost of the *interface* to *value*. The default is 55, the minimum 0, and the maximum 65535.

Description:

The `brconfig` utility is used to configure network bridge parameters and retrieve network bridge parameters and status from `io-pkt`.

A network bridge creates a logical link between two or more IEEE 802 networks that use the same (or “similar enough”) framing format. For example, it's possible to bridge Ethernet and 802.11 networks together, but it isn't possible to bridge Ethernet and Token Ring networks together.

To create a bridge interface, use the *`ifconfig`* (p. 957) command's `create` subcommand. Perform all other bridge configuration using the `brconfig` utility.

Copyright © 2014, QNX Software Systems Limited

Examples:

The following code, when placed in the file `/etc/rc.d/rc.local`, creates a bridge called `bridge0`, adds the interfaces `ray0` and `fxp0` to the bridge, and then enables packet forwarding. You could use such a configuration to implement a simple 802.11-to-Ethernet bridge (assuming the 802.11 interface is in ad hoc mode).

```
ifconfig bridge0 create
brconfig bridge0 add ray0 add fxp0 up
```

Consider a system with two 4-port Ethernet boards. The following code, when placed in the file `/etc/ifconfig.bridge0` causes a bridge consisting of all eight ports to be created with Spanning Tree enabled:

```
ifconfig bridge0 create
brconfig bridge0 \
add tlp0 stp tlp0 \
add tlp1 stp tlp1 \
add tlp2 stp tlp2 \
add tlp3 stp tlp3 \
add tlp4 stp tlp4 \
add tlp5 stp tlp5 \
add tlp6 stp tlp6 \
add tlp7 stp tlp7 \
up
```

Copyright © 2014, QNX Software Systems Limited

bunzip2

Decompress files

Syntax:

```
bunzip2 [options] [files...]
```

Runs on:

QNX Neutrino

Options:

For a complete listing, see *bzip2* (p. 80)

Description:

This utility decompresses files that have been compressed with `bzip2`.

Copyright © 2014, QNX Software Systems Limited

bzcat

Decompress files to standard output

Syntax:

```
bzcat [options] [files...]
```

Runs on:

QNX Neutrino

Options:

For a complete listing, see *bzip2* (p. 80).

Description:

This utility decompresses files that have been compressed with `bzip2`, and sends the output to standard output.

Copyright © 2014, QNX Software Systems Limited

bzip2

Compress and decompress files

Syntax:

Compress and decompress files:

```
bzip2 [options] [files...]
```

Decompress files:

```
bunzip2 [options] [files...]
```

Decompress files to standard output:

```
bzcat [options] [files...]
```

Runs on:

QNX Neutrino

Options:

-1 ... -9

Set the blocksize to 100 ... 900 KB.

-c or --stdout

Send the output to standard output.

-d or --decompress

Force decompression.

-f or --force

Force the utility to overwrite existing output files.

-h or --help

Display a help message.

-k or --keep

Keep (i.e., don't delete) input files.

-L or --license

Display the software version and license.

-q or --quiet

Copyright © 2014, QNX Software Systems Limited

Suppress noncritical error messages.

-s or --small

(Small) use less memory (at most 2500 KB).

-t or --test

Test the integrity of the compressed file.

-V or --version

Display the software version and license.

-v or --verbose

Verbose output; a second `v` makes the utility more verbose.

-z or --compress

Force compression.

--fast

An alias for -1.

--best

An alias for -9.

Description:

This utility compresses and decompresses files:

- If invoked as `bzip2`, the default action is to compress.
- As `bunzip2`, the default action is to decompress.
- As `bzcat`, the default action is to decompress to stdout.

If no file names are given, `bzip2` compresses or decompresses from standard input to standard output.

Copyright © 2014, QNX Software Systems Limited

bzip2recover

Recover data from a damaged `bzip2` file

Syntax:

```
bzip2recover [file]
```

Runs on:

QNX Neutrino

Options:

None.

Description:

This utility recovers data from a damaged `bzip2` file.

Copyright © 2014, QNX Software Systems Limited

Chapter 4
C

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`cd`	*cd* (p. 1061) builtin `ksh` command
`cdpub`	*Device Publishers Developer's Guide*
`curl`	*http://curl.haxx.se/docs/manpage.html*

This chapter describes the utilities, etc. whose names start with “C”.

Copyright © 2014, QNX Software Systems Limited

c++filt

Demangle C++ and Java symbols

Syntax:

c++filt_variant [*options*] [*symbol*…]

where *c++filt_variant* depends on the target platform, as follows:

Target platform	*c++filt_variant*
ARMv7	`ntoarmv7-c++filt`
x86	`ntox86-c++filt`

Runs on:

Linux, Microsoft Windows

Options:

See the GNU website at *http://www.gnu.org/*.

Description:

The C++ and Java languages provides function overloading, which means that you can write many functions with the same name (providing each takes parameters of different types). All C++ and Java function names are encoded into a low-level assembly label (this process is known as mangling). The `c++filt` program does the inverse mapping: it decodes (demangles) low-level names into user-level names so that the linker can keep these overloaded functions from clashing.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

calib-touch

Calibrate a touchscreen

You must be `root` to run this utility.

The `calib-touch` utility is a window manager. If you have another window manager started by your startup script, it is best to run `calib-touch` before running your other window manager.

Syntax:

```
calib-touch [options]
```

Runs on:

QNX Neutrino

Options:

`-size=[`*width*`]x[`*height*`]`

Set the size of the viewport (default is fullscreen).

`-pos=[`*x*`],[`*y*`]`

Set the position of the viewport (default is 0, 0).

`-zorder=[`*int*`]`

Set the z-order of the window (default is 0).

`-format=[`*str*`]`

Set the pixel format of the window (default is the pixel format assigned by Screen Graphics Subsystem). The pixel format can be one of the following strings:

- *rgba8888*
- *rgbx8888*
- *rgb565*

`-interval=[`*int*`]`

Set the swap interval (default is 1).

Copyright © 2014, QNX Software Systems Limited

-verbose=[*int*]

Set the verbosity level (default is 0).

-sample-limit=[*int*]

Set the number of samples for each target (default is 10).

-config-file=[*str*]

Set an alternative configuration filename (default is `/etc/system/config/calib.<hostname>`). This calibration tool requires access to write a configuration file. If `/etc/system/config` is on a read-only filesystem, then you will need to specify a different path and filename so that `calib-touch` has permissions to write the configuration file.

-recalibrate

Perform a calibration despite the existence of a calibration file. The existing calibration file will be overwritten. You may need to recalibrate if you make any changes related to the touch device (e.g., changing your display or display resolution).

Description:

The `calib-touch` utility is used to calibrate a touchscreen. Once the touchscreen is successfully configured , the `calib-touch` utility saves a configuration file at `/etc/system/config/calib.$`*hostname*.

Prior to starting the touch calibration, ensure that you have an appropriate `begin mtouch` section in your `graphics.conf` file to define your touch device.

To invoke the calibration tool:

1. Ensure that `screen` is running.
2. Run `calib-touch` from either your startup script or from a shell.
3. Touch the bull's-eye targets on each of the display screens.
4. Touch the **Accept** button to finish calibration.

You can use the **Esc** key to exit the calibration tool at any time.

Examples:

Calibrate for fullscreen:

```
calib-touch
```


Copyright © 2014, QNX Software Systems Limited

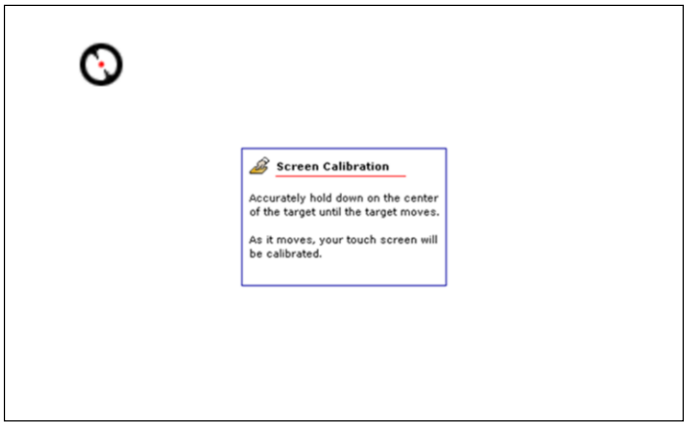


Figure 1: First calibration screen

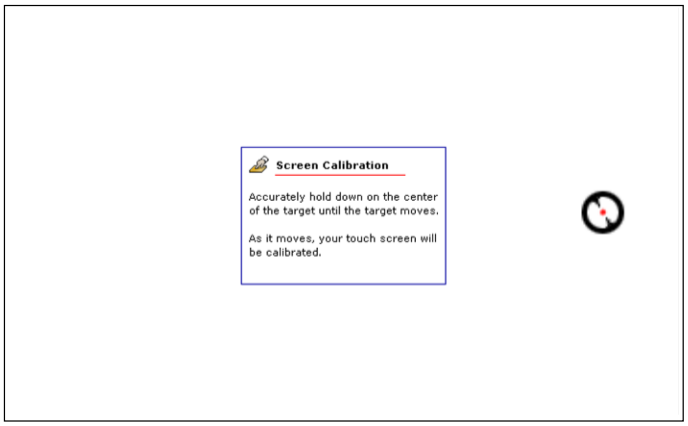


Figure 2: Second calibration screen

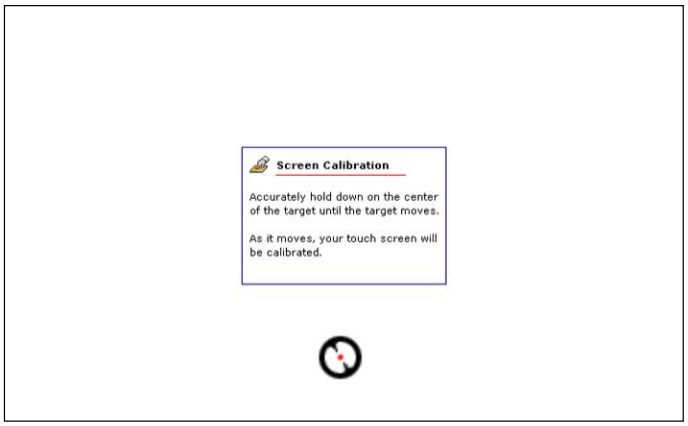


Figure 3: Third calibration screen

Copyright © 2014, QNX Software Systems Limited

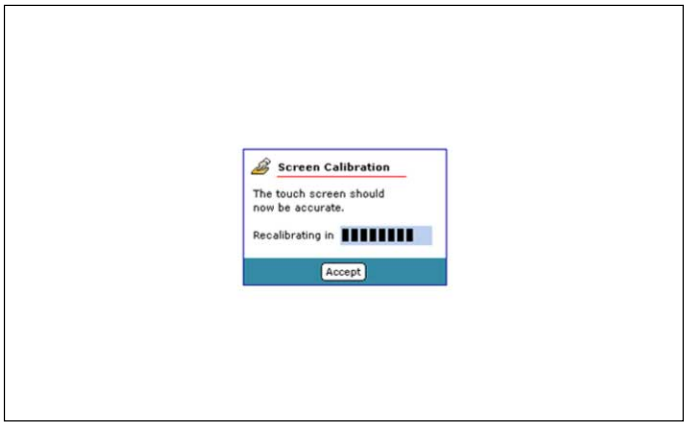


Figure 4: Final calibration screen

Copyright © 2014, QNX Software Systems Limited

cam-cdrom.so

Provide a common access method for CD-ROMs

Syntax:

```
driver ... cdrom cdrom_options ... &
```

Runs on:

QNX Neutrino

Options:

driver

One of the `devb-*` drivers.

The cdrom options control the driver's interface to `cam-cdrom.so` for CD-ROM devices. If specified, they must follow the cdrom keyword:

name=*prefix*

Specify the device prefix (default: `cd`);

retries=*num*@*path_id*:*target*:*lun*

Set the maximum number of command retries on path *N*, target *N*, and LUN *N*. The default is 10.

rmb=truelfalse@*path_id*:*target*:*lun*

Set or clear the device as being removable on path *N*, target *N*, and LUN *N*.

timeout=*g1*:*g2*:*g3*:*rw*@*path_id*:*target*:*lun*

Set the rw and group command timeouts on path *N*, target *N*, and LUN *N*. The defaults are the values from the Timeout & Protect mode page if supported, otherwise 60:90:10:10 (in seconds).

verbose=*verbosity*@*path_id*:*target*:*lun*

Set the verbosity level on path *N*, target *N*, and LUN *N*.

Copyright © 2014, QNX Software Systems Limited

Description:

The `cam-cdrom.so` shared object provides common access methods (CAMs) for CD-ROM devices.

Copyright © 2014, QNX Software Systems Limited

cam-disk.so

Provide a common access method for hard disks

Syntax:

driver ... disk *disk_options* ... &

Runs on:

QNX Neutrino

Options:

driver

One of the devb-* drivers.

The disk options control the driver's interface to cam-disk.so. If specified, they must follow the disk keyword:

name=*prefix*[@*device_number*]

Specify the device prefix and optionally where to start numbering the devices from:

- disk name=usb names the devices /dev/usb0, /dev/usb1, and so on
- disk name=usb@2 starts at /dev/usb2, /dev/usb3, and so on (in the absence of any existing naming conflicts)

Specify device numbers *only* on a closed system where you know all the devices and indexes.

The default is hd.

nobios

Don't use the geometry from BIOS int 13. By default, if you don't specify the translation option, the BIOS geometry is used.

noptab

Don't use the geometry from the partition table. The geometry from the partition table is used if you specify the nobios option, or the BIOS geometry is invalid.

Copyright © 2014, QNX Software Systems Limited

rmb=truelfalse[@*path_id*:*target*:*lun*]

Specify whether or not the device on the given path, target, and LUN is removable.

retries=*num*@*path_id*:*target*:*lun*

Set the maximum number of command retries on the specified path, target, and lun. The default is 10.

timeout=*g1*:*g2*:*g3*:*rw*@*path_id*:*target*:*lun*

Set the rw and group command timeouts on the specified path, target, and LUN. The default values are obtained from the Timeout & Protect mode page if supported; otherwise they're 60:90:10:10 (in seconds).

translation=*heads*[:*sectors*[@*path_ID*[:*target*[:*lun*]]]]

Specify the geometry explicitly; this overrides the geometry from the BIOS and the partition table. The arguments are:

- *heads* and *sectors* — report this many heads (and optionally, sectors) to `io-blk.so` (p. 993) for hard disks (default is 64 heads and 32 sectors). The QNX 4 filesystem doesn't need this information for normal operation. It's needed only to let `fdisk` (p. 734) write the correct boot cylinder for booting.
- *path_ID* — the number of the controller to use, where the first controller is 0 (the default).
- *target* — for IDE, this is the master (0) or slave (1); for SCSI, it's the SCSI ID the device is jumpered for. The default is 0.
- *lun* — the Logical Unit Number for SCSI; not needed for IDE. The default is 0.

verbose[=*level*[@*path_id*:*target*:*lun*]]

Set the verbosity level on the specified path, target, and LUN. If you don't specify a *level*, `cam-disk.so` increments the current level by one.

Description:

The `cam-disk.so` provides common access methods (CAMs) for hard disk devices.

Copyright © 2014, QNX Software Systems Limited

cam-optical.so

Provide a common access method for optical disks

Syntax:

driver ... optical *disk_options* ... &

Runs on:

QNX Neutrino

Options:

driver

One of the devb-* drivers.

The optical options control the driver's interface to cam-optical.so. If specified, they must follow the optical keyword:

name=*prefix*

Specify the device prefix (default: mo);

nobios

Don't use the geometry from BIOS int 13. By default, if you don't specify the translation option, the BIOS geometry is used.

noptab

Don't use the geometry from the partition table. The geometry from the partition table is used if you specify the nobios option, or the BIOS geometry is invalid.

retries=*num@path_id:target:lun*

Set the maximum number of command retries on the specified path, target, and Logical Unit Number (LUN). The default is 10.

rmb=[true|false]@*path_id:target:lun*

Set/clear the device as removable on path *N*, target *N*, and LUN *N*.

timeout=*g1:g2:g3:rw@path_id:target:lun*

Copyright © 2014, QNX Software Systems Limited

Set the rw and group command timeouts on path *N*, target *N*, and LUN *N*. The defaults are the values from the Timeout and Protect mode page if supported, otherwise 60:90:10:10 (in seconds).

translation=*heads*[:*sectors*[@*path_ID*[:*target*[:*lun*]]]]

Specify the geometry explicitly; this overrides the geometry from the BIOS and the partition table. The arguments are:

- *heads* and *sectors* — report this many heads (and optionally, sectors) to `io-blk.so` (p. 993) for hard disks (default is 64 heads and 32 sectors). The QNX 4 filesystem doesn't need this information for normal operation. It's needed only to let `fdisk` (p. 734) write the correct boot cylinder for booting.
- *path_ID* — the number of the controller to use, where the first controller is 0 (the default).
- *target* — for IDE, this is the master (0) or slave (1); for SCSI, it's the SCSI ID the device is jumpered for. The default is 0.
- *lun* — the Logical Unit Number for SCSI; not needed for IDE. The default is 0.

verbose=*verbosity*@*path_id*:*target*:*lun*

Set the verbosity level on path *N*, target *N*, and LUN *N*.

Description:

The `cam-optical.so` provides common access methods (CAMs) for optical disk devices.

Copyright © 2014, QNX Software Systems Limited

cat

Concatenate and print files (POSIX)

Syntax:

```
cat [-c] [-n|-r] [-u] [file...]
```

Runs on:

Microsoft Windows

Options:

-c

Compress sequences of multiple newline characters into single newlines.

-n

Print line numbers, but don't restart the number for each new file.

-r

Print line numbers, restarting the number for each new file.

-u

Write unbuffered output. Data from the input file(s) is written to the standard output without delay as each file is read.

file

The pathname of an input file. If no files are specified, the standard input is used. If a *file* is the dash character (`-`), `cat` reads from the standard input at that point in the sequence.

Description:

The `cat` utility reads files in the order you specify and writes their contents to standard output.

Examples:

Write the contents of `myfile` to standard output:

```
cat myfile
```

Copyright © 2014, QNX Software Systems Limited

Concatenate `doc1` and `doc2` and write the result to `doc.all`:

```
cat doc1 doc2 > doc.all
```

Exit status:

0

All input files were output successfully.

>0

An error occurred.

Caveats:

Because of the shell language mechanism used to perform output redirection, a command such as:

```
cat doc doc.end > doc
```

causes the original data in `doc` to be lost. The file `doc` is opened for write by the shell, and therefore truncated to zero length, before `cat` is executed.

Copyright © 2014, QNX Software Systems Limited

CC, cc

C++, C compilers

Syntax:

```
cc [options] [operands]
CC [options] [operands]
```

Runs on:

Linux, Microsoft Windows

Options:

For a complete list, see *qcc* (p. 1608).

Description:

The cc utility is used to compile code; CC is for compiling C++ code. Under QNX Neutrino, they're both links to qcc.

Under QNX 4, CC and cc are links to an older compiler that you can't use to compile code for QNX Neutrino.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

cfgopen

Search for configuration files (QNX)

Syntax:

`cfgopen` *`config_file`*

Runs on:

QNX Neutrino

Options:

None.

Description:

This utility searches ***`$HOME`***`/.cfg/` for the specified configuration file and displays the file's path if found.

Exit status:

0

The file was found.

1

The file wasn't found.

Copyright © 2014, QNX Software Systems Limited

chat

Automated conversational script with a modem (QNX Neutrino)

Syntax:

```
chat [options] script
```

Runs on:

QNX Neutrino

Options:

-e

Turn on echoing. Echoing may be turned on or off at specific points in the `chat` script by using the ECHO keyword. When echoing is enabled, all output from the modem is echoed to *stderr*.

-f *chatfile*

Read the chat script from *chatfile*. If you use this option, don't also specify a *script* argument. You must have read access to *chatfile*. Multiple lines are permitted in the file. Use spaces or horizontal tab characters to separate the strings.

-r *report file*

Set the file for output of the report strings. If you use the keyword REPORT, the resulting strings are written to this file. If this option isn't used and you still use REPORT keywords, the *stderr* stream is used for the report strings.

-s

Send all log messages from the -v options and all error messages to *stderr*.

-S

Don't use the system log for log messages from the -v options or error messages.

-T *phone_number*

Pass an arbitrary string, usually a phone number, that's substituted for the `\T` substitution meta character in a send string.

-t *timeout*

Copyright © 2014, QNX Software Systems Limited

Set the *timeout* for the expected string to be received. If the string isn't received within the time limit, the reply string isn't sent. An alternate reply may be sent; the script fails if there's no alternate reply string. A failed script causes `chat` to terminate with a nonzero error code.

-U ***phone_number_2***

Pass a second string, usually a phone number, that's substituted for the `\U` substitution meta character in a send string. This is useful when dialing an ISDN terminal adapter that requires two numbers.

-V

Verbose mode, but send all output to *stderr*; `chat` logs all text received from the modem and the output strings that it sends. This device is usually the local console at the station running `chat` or *pppd* (p. 1560).

-v

Verbose mode; `chat` logs all text received from the modem and the output strings that it sends. In order to capture the log messages, you need to have *syslogd* (p. 1885) running.

script

If the script isn't specified in a file with the -f option, the *script* is included as parameters to the `chat` program.

Description:

The `chat` program defines a conversational exchange between the computer and the modem. Its primary purpose is to establish the connection between the Point-to-Point Protocol Daemon (`pppd`) and the remote's `pppd` process.

You should consider the modem functions (*modem_open()*, *modem_read()*, *modem_script()*, and *modem_write()*, described in the *C Library Reference*) as an alternative to `chat`.

Chat script

The chat script defines the communications.

A script consists of one or more “expect–send” pairs of strings, separated by spaces, with an optional “subexpect–subsend” string pair, separated by a dash as in the following example:

```
ogin:-BREAK-ogin: ppp ssword: hello2u2
```

This line indicates that the chat program should expect the string `ogin:`. If it fails to receive a login prompt within the time interval allotted, it's to send a break sequence

Copyright © 2014, QNX Software Systems Limited

to the remote and then expect the string `ogin:`. If the first `ogin:` is received, the break sequence isn't generated.

Once it receives the login prompt, `chat` sends the string `ppp` and then expects the prompt `ssword:`. When it receives the prompt for the password, it sends the password `hello2u2`.

This could be a security issue as only plain-text passwords may be passed.

A carriage return is normally sent following the reply string. It isn't expected in the "expect" string unless it's specifically requested by using the `\r` character sequence.

The expect sequence should contain only what is needed to identify the string. Since it's normally stored on a disk file, it shouldn't contain variable information. It's generally not acceptable to look for time strings, network identification strings, or other variable pieces of data as an expect string.

To help correct for characters that may be corrupted during the initial sequence, look for the string `ogin:` rather than `login:`. The leading `l` character might be received in error and you may never find the string even though it was sent by the system. For this reason, scripts look for `ogin:` rather than `login:` and `ssword:` rather than `password:`.

A very simple script might look like this:

```
ogin: ppp ssword: hello2u2
```

In other words, expect …`ogin:`, send `ppp`, expect …`ssword:`, send `hello2u2`.

In actual practice, simple scripts are rare. At the very least, you should include sub-expect sequences in case the original string isn't received. For example, consider the following script:

```
ogin:--ogin: ppp ssword: hello2u2
```

This is a better script than the simple one used earlier. It looks for the same `login:` prompt, however, if one isn't received, a single return sequence is sent and then it looks for `login:` again. Should line noise obscure the first login prompt, sending the empty line usually generates a login prompt again.

Abort strings

Many modems report the status of the call as a string. These strings may be `CONNECT ED`, `NO CARRIER` or `BUSY`. It's often desirable to terminate the script should the modem fail to connect to the remote. The difficulty is that a script doesn't know exactly which modem string it may receive. On one attempt, it may receive `BUSY`, while the next time it may receive `NO CARRIER`.

Copyright © 2014, QNX Software Systems Limited

These “abort” strings may be specified in the script using the ABORT sequence. It's written in the script as in the following example:

```
ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ATDT5551212
CONNECT
```

This sequence expects nothing and then sends the string ATZ. The expected response to this is the string OK. When it receives OK, chat sends the string ATDT5551212 to dial the telephone. The expected string is CONNECT. If the string CONNECT is received, the remainder of the script is executed. However, should the modem find a busy telephone, it sends the string BUSY. This causes the string to match the abort character sequence. The script then fails because it found a match to the abort string. If it received the string NO CARRIER, it aborts for the same reason. Either string may be received. Either string terminates the chat script.

Report strings

A “report” string is similar to the ABORT string. The difference is that the strings, and all characters to the next control character such as a carriage return, are written to the report file.

The report strings may be used to isolate the transmission rate of the modem's connect string and return the value to the chat user. The analysis of the report string logic occurs in conjunction with the other string processing such as looking for the expect string. The use of the same string for a report and abort sequence is probably not very useful, however, it is possible.

The report strings don't change the completion code of the program.

These report strings may be specified in the script using the REPORT sequence. It's written in the script as in the following example:

```
REPORT CONNECT ABORT BUSY '' ATDT5551212 CONNECT ''
ogin: account
```

This sequence expects nothing and then sends the string ATDT5551212 to dial the telephone. The expected string is CONNECT. If the string CONNECT is received, the remainder of the script is executed. In addition the program writes to the expect-file the string CONNECT plus any characters that follow it, such as the connection rate.

Timeout

The initial timeout value is 45 seconds; you can change it by using the -t option.

To change the timeout value for the next expect string, specify the TIMEOUT string. For exmple:

```
ATZ  OK ATDT5551212 CONNECT TIMEOUT 10 ogin:--ogin:
TIMEOUT 5 ssword: hello2u2
```

This changes the timeout to 10 seconds when it expects the login: prompt. The timeout is then changed to 5 seconds when it looks for the password: prompt.

Copyright © 2014, QNX Software Systems Limited

The timeout, once changed, remains in effect until it's changed again.

Sending EOT

The special reply string of `EOT` indicates that the `chat` program should send an `EOT` character to the remote. This is normally the end-of-file character sequence. A return character isn't sent following the `EOT`. The `EOT` sequence may be embedded into the send string using the sequence `^D`.

Generating break

The special reply string of `BREAK` causes a break condition to be sent. The break is a special signal on the transmitter. The normal processing on the receiver is to change the transmission rate. It may be used to cycle through the available transmission rates on the remote until you're able to receive a valid `login:` prompt. The break sequence may be embedded into the send string using the `\K` sequence.

Escape sequences

The expect and reply strings may contain escape sequences. All of the sequences are legal in the reply string. Many are legal in the expect. The Expect? column in the table below indicates whether or not the sequence is valid in an expect string.

Sequence	Expect?	Description
`''`	Yes	Expects or sends a null string. If you send a null string, it still sends the return character. This sequence may either be a pair of apostrophe or quote characters.
`\b`	Yes	Represents a backspace character.
`\c`	No	Suppresses the newline at the end of the reply string. This is the only method to send a string without a trailing return character. It must be at the end of the send string. For example, the sequence `hello\c` sends the characters `h`, `e`, `l`, `l`, `o`.
`\d`	No	Delay for one second. The program uses `sleep(1)`,

Copyright © 2014, QNX Software Systems Limited

Sequence	Expect?	Description
		which delays for a maximum of one second.
`\K`	No	Insert a BREAK.
`\n`	Yes	Send a newline or linefeed character.
`\N`	No	Send a null character. The same sequence may be represented by `\0`.
`\p`	No	Pause for a fraction of a second. The delay is 1/10th of a second.
`\q`	No	Suppress writing the string to the ***SYSLOG*** file. The string ?????? is written to the log in its place.
`\r`	Yes	Send or expect a carriage return.
`\s`	Yes	Represents a space character in the string. This may be used when it isn't desirable to quote the strings that contains spaces. The sequence `'HI TIM'` and `HI\sTIM` are the same.
`\t`	Yes	Send or expect a tab character.
`\\`	Yes	Send or expect a backslash character.
`\`*ddd*	Yes	Collapse the octal digits *ddd* into a single ASCII character and send that character. (Some characters aren't valid in expect sequences.)

Copyright © 2014, QNX Software Systems Limited

Sequence	Expect?	Description
^C	Yes	Substitute the sequence with the control character represented by C. For example, the character DC1 (17) is shown as ^Q. (Some characters aren't valid in expect sequences.)

Exit status:

0

The normal termination of the program. This indicates that the script was executed without error to the normal conclusion.

1

One or more of the parameters are invalid or an expect string was too large for the internal buffers. This indicates that the program wasn't properly executed.

2

An error occurred during the execution of the program. A read or write operation might have failed for some reason or `chat` might have received a signal such as `SIGINT`.

3

A timeout event occurred when there was an expect string without having a "-subsend" string. This may mean that you didn't program the script correctly for the condition or that some unexpected event has occurred and the expected string couldn't be found.

4

The first string marked as an ABORT condition occurred.

5

The second string marked as an ABORT condition occurred.

6

The third string marked as an ABORT condition occurred.

7

The fourth string marked as an ABORT condition occurred.

Copyright © 2014, QNX Software Systems Limited

...

The other termination codes are also strings marked as an ABORT condition.

Using the termination code, it's possible to determine which event terminated the script. It's possible to decide if the string BUSY was received from the modem as opposed to NO DIALTONE. While the first event may be retried, the second probably has little chance of succeeding during a retry.

Copyright © 2014, QNX Software Systems Limited

chattr

Manipulate the attributes of a file (QNX Neutrino)

Syntax:

```
chattr [+/- attribute]... [filename]...
```

Runs on:

QNX Neutrino

Options:

attribute

The attribute you want to remove (-) or set (+).

filename

The name of a file whose attributes you want to display or manipulate.

Description:

The `chattr` utility is a front end to the `devctl(DCMD_FSYS_FILE_FLAGS)` command (from `<sys/dcmd_blk.h>`), which gets and sets file attributes (that are outside the scope of the POSIX standard). For example, the DOS/FAT filesystem has the concept of "hidden" or "system" files. Such attributes are typically stored as flags in the on-disk inode of each file.

These attributes are divided into the following classes:

Generic

An attribute that corresponds to a concept that may be shared across multiple filesystem formats (even if implemented in a different manner for each one).

Filesystem-specific

An attribute that has meaning only within a specific filesystem.

It's the responsibility of each *io-blk.so* (p. 993) filesystem module to map generic attributes to and from their private representation.

For example, a hidden file is a generic concept of a filename that may be hidden from normal user browsing (not displayed by `ls`) but remains available (to *open()* by an application) if its name is already known. The representation of this concept varies between filesystems:

Copyright © 2014, QNX Software Systems Limited

- For DOS/FAT, it directly corresponds to the “hidden” bit.
- For ISO-9660, it's the “existence” bit.
- The QNX 4 filesystem has no such thing.

This mapping functionality allows for application abstraction away from a particular on-disk structure (a program can work unchanged against any file on any filesystem). Many filesystem-specific attributes have no corresponding generic concept, and must be manipulated with knowledge of the underlying filesystem structure; the various <sys/fs_*.h> header files contain relevant bit definitions.

When you invoke `chattr` with no attributes and a filename (or a list of filenames), it displays the currently set attributes of each file. When you invoke `chattr` with an attribute (or list of attributes), it applies those modifications to each file.

Examples:

List the attributes for a file:

```
# chattr chattr.c
chattr.c: +backup +contiguous +used +modified
```

Add to and remove from the attributes for a file:

```
# chattr -archive +system /fs/dos/autoexec.bat
/fs/dos/autoexec.bat: -archive +system
```

Turn off snapshots for a Power-Safe (*fs-qnx6.so* (p. 823)) filesystem:

```
# chattr -snapshot /fs/qnx6
/fs/qnx6: -snapshot
```


Copyright © 2014, QNX Software Systems Limited

chgrp

Change file group ownership (POSIX)

Syntax:

```
chgrp [-hRv] group file...
```

Runs on:

QNX Neutrino

Options:

-h

Modify the symbolic link instead of the referenced file.

-R

Recursively change group ownership of files. For each *file* that names a directory, `chgrp` changes the group of the directory and of all files in the file hierarchy below it.

-v

Be verbose; display to *stdout* all the operations being performed.

group

A group name from the group database, or a numeric group ID.

file

The pathname of a file whose group ID is to be modified.

Description:

The `chgrp` utility lets you change the group ownership of one or more files. For each file you name, `chgrp` sets the file's group ID to that specified by the *group* argument.

If you invoke `chgrp` with the -R option, and `chgrp` attempts but fails to change the group ID of a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy.

- You must be `root` or the owner of the file in order to change its group ownership. The underlying filesystem might impose further restrictions. For example, the QNX 4 filesystem sets the `_PC_CHOWN_RESTRICTED`

Copyright © 2014, QNX Software Systems Limited

configuration variable; for more information, see *pathconf()* in the QNX Neutrino *C Library Reference*.

- The maximum numeric group ID on a QNX 4 filesystem is 65534.

Examples:

Change the group of `myfile` to 27:

```
chgrp 27 myfile
```

Change the group of `myfile` to `technical`:

```
chgrp technical myfile
```

Files:

/etc/group

This file defines the known group IDs for the system. It associates group names with a numerical ID and a list of users who are members of the group.

Entries in this file appear in the following format:

groupname:*unused*:*groupid*:*user*[,*user*]...

Exit status:

0

The utility executed successfully and all requested changes were made.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

chkdosfs

Check a DOS (FAT-12/16/32) filesystem for consistency (QNX Neutrino)

Syntax:

`chkdosfs [-npuy] *device* | *mountpoint* | *file*`

Runs on:

QNX Neutrino

Options:

-n

Answer "no" to all repair questions and prompts.

-p

Check and fix the filesystem non-interactively (i.e. "preen" mode).

-u

Perform unconditional check of the filesystem (regardless of the on-disk/dirty status).

-y

Answer "yes" to all repair questions and prompts.

device

The device name hosting the DOS filesystem (e.g. `/dev/hd0t6`).

mountpoint

The mountpoint of the DOS filesystem (e.g. `/fs/hd0-dos`).

file

A regular file containing a DOS filesystem image.

Description:

The `chkdosfs` utility performs a consistency check on the specified DOS filesystem. This check consists of multiple passes over the filesystem.

If an error occurs, the action taken depends on the command-line options used. If -p was specified (typically to auto-check the filesystem at startup), no message is displayed

Copyright © 2014, QNX Software Systems Limited

and the default repair action is silently made. If either -n or -y was specified, a descriptive message is displayed and a “no” or “yes” response to the suggested action is automatically generated. Otherwise the user interactively decides on the repair action to make (the suggested default is indicated).

In order to perform repairs, `chkdosfs` requires write access to the device hosting the DOS filesystem. Normally, only `root` has permission for write access; if `chkdosfs` does not have such access, it will still check the filesystem but will operate as if the -n option had been specified.

By default, the `chkdosfs` utility checks an on-disk flag that's maintained by the filesystem that indicates to `chkdosfs` whether or not anything needs to be checked. This flag is usually updated when mounting or unmounting the filesystem. The -u option can be used to force the `chkdosfs` to run regardless of the state of this flag.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

Copyright © 2014, QNX Software Systems Limited

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

Check the filesystem on the DOS partition of a hard disk:

```
# chkdosfs /dev/hd0t11
Phase 1 - Read and compare FATs
Phase 2 - Check cluster chains
Phase 3 - Check directories
Phase 4 - Check for lost files

1476784 kb used, 1010088 kb free, 24932 files, 2921 directories
Filesystem is clean.
```

Exit status:

0

The filesystem was checked and either no errors were detected or all such errors were repaired.

1

The filesystem was not checked. This may be because the user interrupted the operation, a non-recoverable internal error such as insufficient memory occurred, or `chkdosfs` found an unrepairable error.

Contributing author:

Wolfgang Solfrank, Martin Husemann

Copyright © 2014, QNX Software Systems Limited

chkfsys

Check an entire QNX 4 filesystem for consistency (QNX)

You must be `root` to run this utility.

Syntax:

When running on QNX 4:

```
chkfsys [-fpPqrsuvV] [-z zapfile] drive
```

When running on QNX Neutrino:

```
chkfsys [-fpPqrsuvVx] [-z zapfile] mountpoint
```

Or:

```
chkfsys [-fpPqrsuvVx] [-z zapfile] -m drive
```

Runs on:

QNX Neutrino

Options:

-f

Don't fix anything.

-m

(QNX Neutrino only) No mountpoint; the path specified is a raw device/partition.

-p

Prompt before starting.

-P

Suppress prompting (i.e. fix without asking any questions).

-q

Be quiet.

-r

Rebuild the bitmap without prompts or messages.

Copyright © 2014, QNX Software Systems Limited

-s

Suppress the display of statistics.

-u

Check the filesystem, regardless of the status recorded on the disk.

-v

Be verbose. (Shows files in addition to directories as they're being checked. Slows `chkfsys` considerably.)

-V

Very verbose display.

-x

(QNX Neutrino only) Exit with detailed error codes; see below.

-z zapfile

Record pathnames of files that need to be zapped in the specified file. The *zapfile* must reside on a different filesystem from the one being checked.

drive

The disk to check (e.g. `/dev/fd0`, `/dev/hd0t77`).

mountpoint

(QNX Neutrino only) The filesystem mountpoint of the drive (e.g. `/`).

Description:

The `chkfsys` utility performs a consistency check of a QNX 4 filesystem on the requested drive. The `chkfsys` utility doesn't operate on disk partitions containing non-QNX filesystems (e.g. DOS partitions, QNX 2 partitions). In addition, `chkfsys` must have access to the block special file that the filesystem is contained in. For this reason, `chkfsys` can't be used on NFS-mounted QNX filesystems.

The `chkfsys` utility claims that a Power-Safe (*`fs-qnx6.so`* (p. 823)) filesystem is corrupt; use *`chkqnx6fs`* (p. 121) instead.

For QNX filesystems, `chkfsys` recursively walks the filesystem, checking every file on the disk. During the walk, checks are made on the directory entry of each file and the extents that make up the file. A bitmap is constructed in memory that's consistent with the block allocation of all files and directories on the disk. This bitmap is then compared to the existing one on the disk. If they differ, the user is given the option of replacing the existing bitmap on disk with the one constructed in memory.

Copyright © 2014, QNX Software Systems Limited

By default, chkfsys checks an on-disk flag that's maintained by the filesystem that indicates to chkfsys whether or not anything needs to be checked. If the flag is set, chkfsys reports that everything is fine and exits immediately. When you do an orderly shutdown of the system, this flag is always set (unless an error had occurred in the process). If you shut down the system by powering down, the flag may or may not be set, depending on the state of the filesystem at the time. You can use the -u option to force chkfsys to run even if the flag is set.

You should use chkfsys only when the filesystem is stable. There should be NO files open for writing when chkfsys is running. If you make any repairs, remount the filesystem by slaying and restarting the disk driver.

If you aren't doing any fixes (with the -f option), you may check a filesystem with open files, but beware: you may get inconsistent reports in this case.

The chkfsys utility is normally used to recover blocks that were lost through the use of the *zap* (p. 2078) utility. When zap has been used, chkfsys reports that there are blocks used in the bitmap that are in fact *not* used by any file. These blocks may be recovered by writing the reconstructed bitmap back to disk. The chkfsys utility attempts to read each of these blocks, but *doesn't* mark bad blocks as available. Any blocks found this way are added to the /.bad_blks file at the root of the filesystem being checked.

The chkfsys utility tells you if any files are using blocks that are now known to be bad.

If chkfsys reports that a block is used by more than one file, this could indicate one of two problems:

- If bad blocks exist, then this means that the file uses a block that's bad and marked as used in the bitmap.
- If there are no known bad blocks, then a multiple allocation of a single block has occurred.

In either case, the file should be saved on *another* disk (if possible), and the original file should be destroyed with the *zap* (p. 2078) utility. The chkfsys utility should then be run again to update the bitmap, after which the saved file may be restored onto this disk.

In general, whenever the bitmap is replaced, chkfsys should be run a second time to ensure that the filesystem is indeed consistent. To do so you must specify the -u option.

The -f (no fix) option prevents chkfsys from attempting to make any fixes to the filesystem. The disk isn't opened for write, but only for read. This option lets a user examine the filesystem without requiring other users to stop using the disk or filesystem. Beware, however, that the -f option may report errors that don't really exist (if other

Copyright © 2014, QNX Software Systems Limited

users are opening, closing, or growing files during the time that chkfsys is running). Even so, this can be a valuable option for sites that are up and running 24 hours a day, when the system operator carefully evaluates the results. If you see errors that you believe result from current activity, run chkfsys again with the -f option to verify the errors. If you have located errors that require fixing, you should idle the filesystem and run chkfsys without the -f option.

The -p (pause) option is used primarily with floppy disks. You can start chkfsys from a floppy diskette, wait for chkfsys to pause, remove the current disk (which contains the chkfsys command), and then insert another disk you wish to check.

The -q (quiet) option suppresses the display of each filename as that file is checked. This speeds up the checking significantly, without loss of information, because chkfsys shows you the name of any files that have errors.

The -r (rebuild) option suppresses the warning message that normally appears at the end of a chkfsys run when the existing bitmap differs from the newly constructed bitmap in memory. When -r is specified, chkfsys automatically rebuilds the bitmap. Note that this option isn't effective with the -f (no fix) option.

The -s (no stats) option prevents the display of the statistics message that normally appears at the end of a chkfsys run.

The -v (verbose) option displays information on the checking.

The -P (no prompt) option causes chkfsys to automatically fix problems encountered without prompting the user before each fix. However, there are some serious errors (disk IO error or corruption of a high level directory) for which the fix may be to remove a directory (and all its hierarchy/contents). This may not be a prudent action to undertake without user confirmation. In such a situation -P will print an error message and exit. If you wish chkfsys to continue unattended in all circumstances, you can specify this as "-PP".

The -z *zapfile* option is used to record, in the named file, the names of files that should be zapped after chkfsys is finished. The *zapfile* must be on a different filesystem from the one being checked. When a file is found to use an area of the disk allocated to another file, or when a file uses an area of the disk marked as bad in the bitmap, the files must be zapped and chkfsys run again.

After a power failure

The chkfsys utility may also be run after a system crash or power failure, which may have left some files busy. The utility makes the files "unbusy" and also makes checks to ensure that no damage to the filesystem has occurred. QNX is designed to be immune to this type of damage.

In the event of the loss of a filesystem due to the corruption of the root directory and the bitmap (first few blocks of the disk), you should refer to the QNX Neutrino *System Architecture*, and the *dinit* (p. 626) utility documentation for methods of initializing

Copyright © 2014, QNX Software Systems Limited

just those portions of your disk. If only the root block and bitmap are damaged, `chkfsys` is able to recover the files in most cases. If the root directory or inode file is damaged (the next areas on the disk), recovery may be possible with the *`dinit`* (p. 626), *`spatch`* (p. 1830), and `chkfsys` utilities. Note that such repair requires intimate knowledge of the filesystem structure. Many users would just recover lost files from a backup at this point. You shouldn't expect to run into such problems; these are very rare events that we've tried to anticipate.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
7	Windows NT[a]	*`fs-nt.so`* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
77, 78, or 79	QNX 4	*`fs-qnx4.so`* (p. 820)	*`dinit`* (p. 626)	*`chkfsys`* (p. 114)
131	Linux (Ext2)	*`fs-ext2.so`* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*`fs-mac.so`* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*`fs-qnx6.so`* (p. 823)	*`mkqnx6fs`* (p. 1274)	*`chkqnx6fs`* (p. 121)[b]
	Read-only compressed (RCFS)	*`fs-rcfs.so`* (p. 827)	*`mkrcfs`* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

Check the filesystem on the QNX partition of a hard disk:

```
chkfsys /hd
```


Copyright © 2014, QNX Software Systems Limited

Check the QNX filesystem mounted as the root (/) and automatically rebuild the bitmap:

```
chkfsys -rs /
```

Exit status:

The exit status depends on whether or not you specified the -x option:

- If you didn't specify the -x option, the exit status is as follows:

  0

  The filesystem(s) have been checked.

  An exit status of zero *doesn't* indicate that no problems were found with the filesystem(s). It merely indicates that no irrecoverable errors internal to the `chkfsys` utility were encountered.

  >0

  The filesystem(s) may not have been checked. The `chkfsys` operation may have been interrupted at the request of the user or an internal error (such as running out of memory) may have occurred.

- If you specified the -x option, the lower bit of the error code indicates whether or not anything was fixed, while the upper four bits identify the cause of failure if nonzero:

  0x00

  The filesystem was clean; there was nothing to do.

  0x01

  The disk data has been fixed.

  0x10

  Insufficient memory.

  0x20

  General program failure.

  0x30

  The filesystem query or request failed.

  0x40

Copyright © 2014, QNX Software Systems Limited

Failed to read from the device.

0x50

Failed to write to the device.

0x60

Filesystem corruption was detected but not fixed.

Copyright © 2014, QNX Software Systems Limited

chkqnx6fs

Check an entire Power-Safe filesystem for consistency (QNX Neutrino)

You must be logged in as `root` to run this utility.

Syntax:

```
chkqnx6fs [-fsv] [-S block_number] host
```

Runs on:

QNX Neutrino

Options:

-f

Fix the filesystem unconditionally. The default is to check only.

-S *block_number*

Force the display of superblock 0 or 1 (implies -s and -vv).

-s

Display header information from the superblock. The number of -v options controls which fields `chkqnx6fs` displays.

If you specify -s, `chkqnx6fs` locates and verifies the active superblock, but doesn't check the filesystem itself.

-v

Increase output verbosity. You can specify multiple -v options.

host

The host of the filesystem. You can specify this as a block-special device or partition (e.g. `/dev/hd0t76`), as a regular file, or as the root directory of a mounted `fs-qnx6` filesystem (which will be resolved to the real host device).

Copyright © 2014, QNX Software Systems Limited

Description:

The `chkqnx6fs` performs a consistency check on a Power-Safe (*`fs-qnx6.so`* (p. 823)) filesystem. The check is conducted in these passes:

1. Locate and verify superblocks and select the newest stable one.
2. Traverse the system inodes and bitmap files.
3. Recursively walk the directory hierarchy from the root.

Only a stable filesystem can be checked (i.e. one that isn't going to change or be updated during the scan). A stable filesystem is one of the following:

- unmounted
- mounted read-only
- mounted read-write with snapshots disabled (in which case the stable filesystem, not the working copy, is checked)

You shouldn't actually need to use `chkqnx6fs` in a production system (e.g. in a boot script). The design of the `fs-qnx6` filesystem should (in the absence of software bugs, physical bad blocks, or malicious data modification on the raw device) make any such check unnecessary.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
7	Windows NT[a]	*`fs-nt.so`* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
77, 78, or 79	QNX 4	*`fs-qnx4.so`* (p. 820)	*`dinit`* (p. 626)	*`chkfsys`* (p. 114)
131	Linux (Ext2)	*`fs-ext2.so`* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*`fs-mac.so`* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*`fs-qnx6.so`* (p. 823)	*`mkqnx6fs`* (p. 1274)	*`chkqnx6fs`* (p. 121)[b]

Copyright © 2014, QNX Software Systems Limited

Partition type	Filesystem	Shared object	Initialize with:	Check with:
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

```
# chkqnx6fs -v /dev/hd0t76
** Prelude: read and verify superblocks **
** Pass 1 : verify bitmap and inodes **
** Pass 2 : verify directory hierarchy **
** Summary: 20216/8040524 blocks, 142/62816 inodes **
```

Exit status:

0

The filesystem is consistent/stable.

1

An error occurred checking the filesystem (descriptive messages are written to *stderr*).

Copyright © 2014, QNX Software Systems Limited

chmod

Change file modes (POSIX)

Syntax:

```
chmod [-Rv] mode file...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-R

Recursively change file modes. For each *file* that names a directory, `chmod` changes the file mode bits of the directory and all files in the file hierarchy below it.

-v

Be verbose; display the operations that are being performed.

mode

Represents the change to be made to the file mode of each *file* named (see the Description below).

file

The pathname of a file whose file mode bits are to be modified.

Description:

The `chmod` utility lets you change any or all of the file permission mode bits of one or more files. For each file that you name, `chmod` changes the file permission mode bits according to the *mode* operand.

To change a file's permission mode bits, the user of `chmod` must be either the owner of the file or the superuser, `root`.

The *mode* option can be either a *symbolic_mode* expression or a nonnegative octal integer.

Changing the permissions may change any access control lists associated with the file or directory. For more information, see “Access Control Lists (ACLs)” in the QNX Neutrino *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

Symbolic Modes

The *symbolic_mode* has the following form:

```
[who]operator[copy|permissions][,symbolic_mode]
```

The *who* part of the symbolic mode is any combination of:

a

User, group, and other access

g

Group access

o

Other access

u

User access

The *operator* is one of:

+

Add specified permissions to the group, other, or user category of the specified files.

-

Remove specified permissions from the group, other, or user category of the specified files.

=

Set the specified permissions for the group, other, or user category of the specified files.

The *copy* part specifies the unmodified permissions (i.e. before the chmod command has been executed) of one of:

g

Group

o

Other

u

User

Copyright © 2014, QNX Software Systems Limited

The *permissions* part is any combination of:

r

Read permission

s

When executed, set the user ID (if *who* contains or implies u) and/or group ID (if *who* contains or implies g)

t

Sticky bit

w

Write permission

X

Execute/search permission if the file is a directory or at least one execute bit is on before any of the file mode bits are modified

x

Execute permission

The *who* specification is optional. When it isn't supplied, all the permissions (user, group and other) are affected, but for + and = operators, only those permissions that aren't set in the file creation mask (see *umask* (p. 2005)) are set.

The *permissions* part is also optional. If omitted, it defaults to none (i.e. the command adds no permissions, removes no permissions, or sets the permissions to none, depending on the operator).

Some examples of symbolic modes:

Make `myfile` executable by all:

```
chmod a+x myfile
```

Remove read permission for group and others:

```
chmod og-r myfile
```

Perform both the above operations, in the order given, on three files: `myfile`, `file2`, and `zzz`:

```
chmod a+x,og-r myfile file2 zzz
```

Add read permission to the user, remove write permission from the user, and set the group permissions to be the same as the other permissions:

```
chmod u+r,u-w,g=o myfile
```


Copyright © 2014, QNX Software Systems Limited

Octal Modes

In octal mode, permissions are specified with a three-digit octal number. The three digits represent user, group, and other permissions in that order.

Each permission may be specified with an octal number: read = `4`; write = `2`; execute = `1`; no permission = `0`. The octal equivalents are derived by adding the numbers associated with the four basic permissions. The following table illustrates their use:

Octal number	Symbolic	Permission
`0`	`---`	None
`1`	`--x`	Execute
`2`	`-w-`	Write
`3`	`-wx`	Write/execute
`4`	`r--`	Read
`5`	`r-x`	Read/execute
`6`	`rw-`	Read/write
`7`	`rwx`	Read/write/execute

For example, give the user read/write/execute (octal `7` = `rwx`), group read/execute (octal `5` = `r-x`), and other read only (octal `4` = `r--`) for the file `myfile`:

```
chmod 754 myfile
```

Setgid and setuid

The following table shows how the setgid and setuid file modes are represented in octal:

Octal number	File mode
1000	Sticky
2000	Setgid
4000	Setuid
6000	Setgid and setuid

You can combine these file modes with the permission modes described above. For example:

```
chmod 4666 testfile
```

In this case, setuid is set, and the user, group, and other get read/write access.

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

The utility executed successfully and all requested changes were made.

>0

An error occurred.

Caveats:

If the *mode* operand isn't valid, `chmod` doesn't change the file mode bits of any file.

Copyright © 2014, QNX Software Systems Limited

chown

Change the ownership of files and directories (POSIX)

Syntax:

```
chown [-hRv] owner[:group] file...
```

Deprecated:

```
chown [-hRv] owner[.group] file...
```

Runs on:

QNX Neutrino

Options:

-h

Modify the symbolic link instead of the referenced file.

-R

Recursively change ownership of files. For each *file* operand that names a directory, chown changes the user ID of that directory and of all files in the file hierarchy below it.

-v

Verbose. Display to *stdout* the operations which are being performed.

owner

A username from the user database, or a numeric userid. The chown utility changes the owner of each *file* to the user ID of the specified owner.

group

A group name from the user database, or a numeric group ID. The chown utility changes the group of each *file* to the group ID of the specified group.

file

The pathname of a file whose ownership is to be modified.

Description:

The chown utility sets each file's owner and group to the user and group IDs specified by the *owner* and *group* operands.

Copyright © 2014, QNX Software Systems Limited

The maximum numeric group or user ID on a QNX 4 filesystem is 65534.

Examples:

Change the owner of file `data` to user 27:

```
chown 27 data
```

Change the owner of the file `data` to `dtdodge`:

```
chown dtdodge data
```

Change the owner of the file `subfile` to `dtdodge` and set the group of the file to `techies`:

```
chown dtdodge:techies subfile
```

Exit status:

0

The utility executed successfully and all requested changes were made.

>0

An error occurred.

Caveats:

If you invoke `chown` with the -R option, and `chown` attempts but fails to change the owner or group of a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy. The `chown` utility can fail to change the user or group of a file if you don't have appropriate permissions.

In QNX Neutrino, the `_PC_CHOWN_RESTRICTED` flag is enforced, therefore you must be `root` to use `chown` unless you are changing ownership to yourself. Normal users can't give a file away to another user by changing the file ownership.

For compatibility with some other implementations of `chown`, a deprecated syntax allows a period (`.`) to be used instead of a colon (`:`) to separate *user* and *group* (e.g. *user*`:`*group* and *user*`.`*group* are both allowed). However, be aware that if a userid contains a period, it may be specified either alone or in conjunction with a group using `:`, but may not be used in conjunction with a group using `.`. For instance, if there was a userid `my.name` and a group `tech`, you could do a `chown my.name myfile` or `chown my.name:tech myfile`, but *not* `chown my.name.tech myfile`.

Copyright © 2014, QNX Software Systems Limited

cksum

Display file checksums and block counts (POSIX)

Syntax:

```
cksum [-o algorithm] [-q|-v] [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-o algorithm

Use the specified algorithm. Valid *algorithm* values include:

Algorithm	Action
1	Use historic 16-bit checksum algorithm.
2	Use historic 32-bit checksum algorithm.
9	Use 1003.2 draft 9 algorithm (QNX versions 4.0 and 6)
11	Use 1003.2 draft 11 algorithm
12	Use 1003.2 draft 12 algorithm
92	Use 1003.2-1992 standard algorithm (DEFAULT)
4.1	Use old QNX cksum algorithm (QNX 4.10-4.21)

-q

Quiet. Don't display header (counteracts -v). (Default)

-v

Verbose. Display a header which states the algorithm used and names the output columns.

file

Copyright © 2014, QNX Software Systems Limited

The pathname of a file to be checked. If no files are specified, the standard input is used.

Description:

The `cksum` utility writes one line to standard output for each file you specify. This line contains the checksum of the file, as well as the file size and the name of the file being checked. The format of this output varies slightly depending on the command line options specified to `cksum` as follows:

-o algorithm:	Filesize units	Output format
1	Kilobytes	`%lu %lu %s`
2	512-byte blocks	`%lu %lu %s`
All others	Bytes	`%10lu %10lu %s`

If you don't specify any files, `cksum` processes standard input; no filename is given in the output line.

The `cksum` utility lets you quickly compare a suspect version of a file to a trusted version of the same file. You can also use `cksum` to check files after they have been transferred by modem, restored from backup media, or unpacked from a compressed form. The utilities that perform these operations have their own checks, but `cksum` serves as a useful independent checking mechanism.

If you wish to perform a byte-by-byte comparison of files, you can use the *cmp* (p. 134) utility.

Exit status:

0

All files were processed successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

clear

Clear the screen

Syntax:

```
clear
```

Runs on:

QNX Neutrino

Options:

None.

Description:

This utility clears the text-mode screen.

Copyright © 2014, QNX Software Systems Limited

cmp

Compare two files (POSIX)

Syntax:

```
cmp [-l|-s] file1 file2
```

Runs on:

QNX Neutrino

Options:

-l

("el") Print the byte position (in decimal) and the differing bytes (octal) for all differences (not just the first one) between the two files.

-s

Be silent. Return the exit status only.

file1

The pathname of the first file to be compared. If *file1* is the dash (-) character, standard input is used.

file2

The pathname of the second file to be compared.

Description:

The cmp utility compares two files.

This utility is intended for comparing binary files, if you want to compare text files, use *diff* (p. 619).

If you don't specify any options, cmp behaves as follows:

- If the two files are the same, cmp writes no output.
- If the files differ, cmp writes to standard output the byte and line number at which the first difference occurred. Bytes and lines are numbered beginning at 1.

If you specify both the -s and -l options, nothing is printed (no long output).

Copyright © 2014, QNX Software Systems Limited

Examples:

Compare the files `myfile.dat` and `save.dat`:

```
cmp myfile.dat save.dat
```

Exit status:

0

The files are identical.

1

The files differ. This includes cases where one file is identical to the first part of the other. In such cases, if you haven't specified the -s option, cmp writes to standard error a message that EOF was reached in the shorter file (before any differences were found).

>1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

comm

Select or reject lines common to two files (POSIX)

Syntax:

```
comm [-123] file1 file2
```

Runs on:

QNX Neutrino

Options:

-1

("One") Suppress the writing of lines found only in *file1*.

-2

Suppress the writing of lines found only in *file2*.

-3

Suppress the writing of lines found in both *file1* and *file2*.

file1

The pathname of the first file to be compared. If *file1* is `-`, standard input is used.

file2

The pathname of the second file to be compared. If *file2* is `-`, standard input is used.

Description:

The `comm` utility reads *file1* and *file2*, which must be ordered in collating sequence (see the *sort* (p. 1826) utility), and produces three text columns as output.

This column:	Contains:
1	Lines found only in *file1*
2	Lines found only in *file2*
3	Lines found in both files

Copyright © 2014, QNX Software Systems Limited

Examples:

Print only the lines common to both files:

```
comm -12 test.dat save.dat
```

Files:

Standard input is used only if specified as a dash (-) command-line *file* parameter.

All input files must be text files for comm to produce a meaningful result.

The results of the comparison will be written to standard output.

If any errors occur, comm writes diagnostic messages to standard error.

Exit status:

0

All input files were successfully output as specified.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

confstr

Get or set a configuration string

Syntax:

```
confstr [name [value]]
```

Runs on:

QNX Neutrino

Options:

None.

Description:

This utility gets or sets the value of a configuration string. If given, the *name* must be one of the following (in lower- or uppercase):

architecture

The name of the instruction set architecture for this node's CPU(s).

domain

The domain name.

hostname

The name of this node in the network.

A hostname can consist only of letters, numbers, and hyphens, and must not start or end with a hyphen. For more information, see *RFC 952*.

hwprovider

The name of the hardware manufacturer.

hwserial

Serial number associated with the hardware.

libpath

A value similar to the ***LD_LIBRARY_PATH*** environment variable that finds all standard libraries.

Copyright © 2014, QNX Software Systems Limited

machine

This node's hardware type.

path

A value similar to the ***PATH*** environment variable that finds all standard utilities.

release

The current OS release level.

resolve

The contents of the `resolv.conf` file, excluding the domain name.

srpc_domain

The secure RPC domain.

sysname

The operating system name.

version

The current OS version number.

Although these names are similar to the `_CS_*` names accepted by *confstr()*, *getconf* (p. 897), and *setconf* (p. 1745), this utility requires that you specify the names as given above.

If you don't specify a *name*, `confstr` displays the current value of all configuration strings.

To set the value of the configuration string, specify a *value* on the command line.

You must be logged in as `root` in order to set a value.

You can create a symbolic link to `confstr` that corresponds to one of the possible names (e.g. in the `/bin` directory, run `ln -s confstr domain`), and you can then use the symlink to set or get that name value (e.g. `domain` to get the domain, or `domain` *mydomain* to set the domain).

Copyright © 2014, QNX Software Systems Limited

coreinfo

Display information about a core file

Syntax:

```
coreinfo [-ilmstv] file [file ...]
```

Runs on:

QNX Neutrino

Options:

-i

Display process information, including process ID, parent process ID, arguments, initial environment variables, number of threads, and so on.

-l

Display the `QNT_LINK_MAP` note if present. This note includes a link map of shared memory objects, along with build IDs of the associated binaries.

-m

Display the memory map.

-s

Display system information.

-t

Display information about thread(s).

-v[v...]

Be verbose. Specifying additional -v options increases verbosity.

Description:

The `coreinfo` utility displays information about a core file. It lets you examine a core dump directly, without using a debugger.

You can use the options to specify which sections of the core file to display. If you don't specify any options, `coreinfo` displays all the sections.

Copyright © 2014, QNX Software Systems Limited

cp

Copy files (POSIX)

Syntax:

```
cp [-f|-i] [-ABcDNnpqstuVvXx] [-l n] source_file target_file

cp [-f|-i] [-ABcDNnpqstuVvXx] [-l n] [-M qnx|unix]
source_file... target_dir

cp -R [-H|-L|-R] [-f|-i] -R [-ABcDNnpqstuVvXx] [-l n] [-M
qnx|unix] source_file... target_dir

cp -r [-H|-L|-R] [-f|-i] -R [-ABcDNnpqstuVvXx] [-l n] [-M
qnx|unix] source_file... target_dir
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-A

(QNX Neutrino extension) Preserve source file access times.

-B

(QNX Neutrino extension) Use a very small (2 KB) copy buffer.

-c

(QNX Neutrino extension) Create any directories necessary to open the destination path. For example, if the directory `/home/eric` doesn't exist, and you enter:

```
cp -c file /home/eric/source/file
```

`cp` performs the equivalent of:

```
mkdir -p /home/eric/source
cp file /home/eric/source/file
```

-D

(QNX Neutrino extension) Descend past device boundaries when using the -R option. This is the default behavior; if you want to prevent `cp -R` from descending past device boundaries, use the -N option.

-f

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino extension) Force the unlinking of the destination file prior to copying. This option prevents interactive prompting (unless you also specify -i) but *doesn't* disable diagnostic messages.

-H

(QNX Neutrino 6.6 or later) Follow symbolic links in source operands. Symbolic links found in tree traversal aren't followed.

-i

Run interactively; always prompt for confirmation when the destination path exists, regardless of whether you have write permission for the destination file. The -i option is useful when you want to avoid accidentally clobbering files when copying. When you don't have write permission for the destination file and you answer yes to the prompt, the destination file is unlinked first. Otherwise, the destination is simply overwritten and truncated.

The combination of -i and -f works as if you specified only -i, except that when you answer `yes` to the prompt, the destination is always unlinked first — even if you have write permission for the destination file. When you specify only -i, the destination is unlinked only when you don't have write permission for the destination file.

-L

(QNX Neutrino 6.6 or later; the former -L option is now -q) Follow symbolic links.

-l *n*

("el" — QNX Neutrino extension) If *source_file* is a directory, and you specify the -R or -r option, copy only *n* levels of the directory tree. If you specify -l 0, -R or -r is defeated; only files at the current level (files named directly on the command line) are copied.

-M qnx|unix

(QNX Neutrino extension) Do recursive copies in UNIX (the default) or old-style QNX mode.

QNX has, in the past, copied the **contents** of the directories named on the command line into the target directory. UNIX copies the directory itself into the target directory (like `mv`). In either case, if there's only one directory being copied and the destination names a directory that doesn't exist, `cp` creates the destination directory and then copies the **contents** of the source directory into the destination directory.

Copyright © 2014, QNX Software Systems Limited

The default mode under QNX Neutrino is different from that under QNX 4.

For more information, see "*Recursive Copying* (p. 146)," below.

-N

(QNX Neutrino extension) Don't descend past device boundaries when using the -R option. By default, `cp -R` descends past device boundaries while traversing a directory tree; specifying -N prevents this behavior. For example:

```
cp -R / /hd/backup
```

causes `cp` to back up the contents of the disk, including the contents of the `/dev` directory.

In this particular example, only the disk devices (block special files) actually have their data backed up to files in `/hd/backup/dev` because `cp` doesn't copy character special files on recursive copies.

The addition of -N, as follows:

```
cp -RN / /hd/backup
```

causes the contents of the disk to be backed up, but the `/dev` directory is skipped, since it doesn't exist on the hard disk device.

-n

(QNX Neutrino extension) See the -u option.

-P

(QNX Neutrino 6.6 or later) Don't follow symbolic links.

-p

After copying, attempt to duplicate the modification time and file mode of each input file in the corresponding output file. Also duplicate the ownership of each file if the process is run with the privileges of the superuser (`root`). If the process doesn't have the appropriate privileges, the duplication fails.

-q

(QNX Neutrino 6.6 or later; QNX Neutrino extension) Attempt to preserve hard links. When `cp` encounters a file that has a link count greater than 1, it remembers that file's device ID and serial number (inode). If during the `cp` another file with a link count greater than 1 is found matching the serial number and device ID, `cp` creates a link instead of making a second copy of the file. When the destination media is changed, `cp` wipes its memory of

Copyright © 2014, QNX Software Systems Limited

links encountered to that point. (This is significant when making floppy backups, or backups to removable hard disks.)

Before QNX Neutrino 6.6, this option was -L.

-R

If the *source_file* is a directory, recursively copy the directory with the files and subdirectories under it, attempting to preserve special files. The QNX Neutrino RTOS doesn't allow block special files and character special files to be created in this manner. Read the section on *recursive copying* (p. 146), and see the -M and -r options.

-r

Recursively copy directories. If a source file is a special file (e.g., FIFO, named special file), cp doesn't create a special file as the destination. Read the section on *recursive copying* (p. 146) and see the -M and -R options.

-s

(QNX Neutrino extension) Run safely; copy only if the existing destination file has write permission. If the file doesn't have write permission, skip the file without prompting.

-t

(QNX Neutrino extension) Don't attempt to duplicate file time and mode if the -p option was specified or if the ***POSIX_STRICT*** environment variable is set to on.

-u

(QNX Neutrino extension) Copy only if the source is newer than the destination (i.e., the source has a more recent file-modified time), or if the destination doesn't already exist.

-V

(QNX Neutrino extension) Be extra verbose. In addition to doing everything -v does, this option displays a running progress counter (% complete) and it also displays lines when cp skips a file or a directory (i.e. you can see what cp isn't doing as well as what it is doing).

For example, if you select options -R and -n, you'll find that cp -VRn is more useful than cp -vRn, because the -v option in this case might let cp go away and put you back at the prompt without providing you with any feedback.

Copyright © 2014, QNX Software Systems Limited

-v

(QNX Neutrino extension) Be verbose. Display a line of explanatory text every time a file is copied or a directory is created.

-X

(QNX Neutrino extension) Copy only if the destination file doesn't exist.

-x

(QNX Neutrino extension) Copy only if the destination file already exists.

source_file

The pathname of a file to be copied. If you want *source_file* to name a directory, you must also specify the -R option.

target_file

The pathname to which a single file is copied.

target_dir

The pathname of an *existing* directory that's to contain the output file(s).

Description:

There are two syntax forms for cp:

cp [*options*] *source_file target_file*

The cp utility copies the contents of the source file to the destination file named by *target_file*. This first syntax form is assumed when the destination file isn't an existing directory and there's only one source file.

cp [*options*] *source_file... target_dir*

For each *source_file*, cp copies the contents of the file to a destination file in the existing directory named by *target_dir*. The destination's filename under the target directory is the same as its basename (final path component), unless it's a directory (see "*Recursive Copying* (p. 146)"). For example:

```
cp dir/dir/myfile /existingdir
```

copies the contents of `dir/dir/myfile` to the file `/existingdir/myfile`.

This second form is assumed when the destination file is an existing directory or when you specify more than one source file.

Copyright © 2014, QNX Software Systems Limited

If a source file has an access control list (ACL), `cp` doesn't copy it to the destination, but if the destination file already exists and has an ACL, its ACL is preserved.

General

Unless you specify the -R (recursive) option, `cp` refuses to copy any *source_file* that is a directory.

For duplicating lists of files, see the `pax -rw` utility, which is another POSIX utility for duplicating files. You can select sets of files that match complex criteria by using *`find`* (p. 750), and then pipe them to *`pax`* (p. 1444).

What `cp` does when a destination file already exists depends on the options used. If you don't specify -f or -i, `cp` prompts you only if you don't have write permission for the existing destination file. When this happens, you're asked if you want to unlink the file first. If you don't, `cp` goes on to any remaining files. You're prompted only if *stdin* is a tty. Otherwise, `cp` prints a diagnostic message to *stderr* and skips that file.

If you're copying to removable media, such as a floppy or removable disk, and the media becomes full, `cp` closes and removes the incompletely copied destination file, displays a message, then exits.

Recursive copying

When doing a recursive copy of a directory, the destination must be a directory. If you're copying more than one item, the directory must already exist. If you're copying a single directory, `cp` creates the destination directory (all intermediate directories must already exist unless you specify the -c option).

There are two recursive copying modes available with `cp`:

- In the historical QNX 4 behavior, specified by the -Mqnx option, `cp` copies the files and directories underneath the source directory to the destination directory. The source directory itself isn't duplicated within the destination directory.
- The default mode (-Munix) causes `cp` to duplicate the source directory within the destination directory (unless a single directory is being copied and the destination directory doesn't yet exist, in which case -Munix and -Mqnx modes do the same thing).

The default mode under QNX Neutrino is different from that under QNX 4.

In the default -Munix mode, `cp -r /bin /mydir/bin` duplicates `/bin` in `/mydir/bin`, i.e. the destination is `/mydir/bin/bin` and, for example, the file

Copyright © 2014, QNX Software Systems Limited

`/bin/sh` is copied to `/mydir/bin/bin/sh`. This is analogous to the way the *mv* (p. 1328) utility treats destinations.

In -Mqnx mode, `cp -Mqnx -r /bin /mydir/bin` copies the contents of `/bin` to `/mydir/bin` (so, for example, `/bin/sh` is copied to `/mydir/bin/sh`).

Examples:

Copy `file1`, `file2`, and `file3` from the current working directory to the `/home/eric` directory:

```
cp file1 file2 file3 /home/eric
```

Perform a backup of the entire contents of the `home` directory to floppy disks (assuming that `/f0` is a mountpoint for `/dev/fd0`), in the (default) UNIX recursive-copy mode:

```
cp -rvp /home /f0
```

Do the same in QNX recursive-copy mode:

```
cp -Mqnx -rvp /home /f0/home
```

Recursively copy the `/home/eric` directory to the `/home/ejohnson` directory, assuming `/home/ejohnson` doesn't yet exist (this works in either -Munix or -Mqnx mode):

```
cp -rv /home/eric /home/ejohnson
```

Do the same in -Mqnx mode if the directory `ejohnson` already exists:

```
cp -Mqnx -rv /home/eric /home/ejohnson
```

Do the same in -Munix mode if the directory `ejohnson` already exists:

```
cp -Munix -rv /home/eric/. /home/ejohnson
```

Recursively copy the contents of the current directory into `/mydir/` in -Mqnx *or* -Munix mode:

```
cp -Rpv . /mydir/
```

Do the same in -Munix mode only:

```
cp -Munix -Rpv * /mydir/
```

Using -Mqnx instead of -Munix in the previous example copies the *contents* of the directories named on the command line into `/mydir/` (i.e. the file `./bin/ls` is copied to `/mydir/ls`, and the directory `./usr/bin` is `/mydir/bin` in the destination).

Copyright © 2014, QNX Software Systems Limited

Recursively copy the `/home/eric` directory to the `/backups/eric` directory:

```
cp -rv /home/eric /backups
```

Do the same in QNX-style recursive copy mode:

```
cp -Mqnx -rv /home/eric /backups/eric
```

Files:

Input files

If you don't specify the -r option, and you name only one source file, that source file may be of any filetype.

If you specify the -r option, or there's more than one source file, the input files specified by each *source_file* operand, including those files contained within named directories, must be either regular files, block special files, or directories.

If you use the -R option, FIFOs are duplicated in the destination directory structure, but contents of the source FIFOs aren't copied. If `cp` encounters any block special or character special files in the source files, an error occurs, because `cp` can't create them at the destination.

Output files

Each newly created output file is one of the following:

- A directory that contains copies of the files and subdirectories — if any — found in the input directory.
- A regular file that has the same contents as the corresponding input file.
- A FIFO that was created because the corresponding input file was a FIFO and you specified -R. The data from the original FIFO *isn't* copied into the new FIFO (i.e. the new FIFO is empty).
- A symbolic link that was created because the corresponding input file was a symbolic link and you specified -R. The new symbolic link references the same pathname as the original symbolic link.

If an existing destination names some other type of file, `cp` opens it for writing and attempts to copy the contents of the corresponding input file to it.

Environment variables:

POSIX_STRICT

Affects whether file modification times are copied, and, if set on, causes the QNX Neutrino extension options to be disabled.

Copyright © 2014, QNX Software Systems Limited

The setting of the ***POSIX_STRICT*** environment variable affects the -p and -t options, as follows:

POSIX_STRICT	Option	Action
Set	Neither -p nor -t	If the destination doesn't exist, duplicate the mode only.
Set	-p	Duplicate the time and mode; if run by `root`, also duplicate the user ID and group ID.
Set	-pt	If run by `root`, duplicate the user ID and group ID.
Set	-t	If destination doesn't exist, duplicate the mode only.
Unset	Neither -p nor -t	Duplicate the time and mode.
Unset	-p	Duplicate the time and mode; if run by `root`, also duplicate the user ID and group ID.
Unset	-pt	If run by `root`, duplicate the user ID and group ID.
Unset	-t	If destination doesn't exist, duplicate the mode only.

Exit status:

0

All input files were copied successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Caveats:

If cp is copying multiple files or doing a recursive copy, but you *didn't* specify the -R option, cp refuses to copy FIFO and character special files.

If you specify the -R option, and cp attempts but fails to copy a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy.

Copyright © 2014, QNX Software Systems Limited

cpio

Copy file archives in and out (UNIX)

Syntax:

Read/list an archive:

```
cpio -i[Bcdfmrtuv] [pattern...]
```

Write an archive:

```
cpio -o[Bacv]
```

Copy files:

```
cpio -p[adlmruv] directory
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Reset access times of input files after they've been copied. When the -l option is also specified, the access times of linked files aren't reset. You can use this option only with the -o or -i options.

-B

Cause input/output to be blocked 5120 bytes to the record. You can use this option only with the -o or -i options for data directed to or from character special files.

-c

Write header information in ASCII (Default; option present for compatibility)

-d

Create directories as needed. You can use this option only with the -i or -p options.

-f

Copy in all files except those in patterns. You can use this option only with the -i option.

-i

Copyright © 2014, QNX Software Systems Limited

Copy in. (Extract files from an archive being read from standard input.)

-l

("el") Whenever possible, link files rather than copy them. You can use this option only with the -p option.

-m

Retain previous modification times. This option won't work on directories that are being copied. You can use this option only with the -i or -p options.

-o

Copy out. (Write an archive to standard output.)

-p

Pass. Conditionally copy files from a list read from standard input to the destination directory named as an argument to `cpio`.

-r

Interactively rename files. A new name for each file is requested from the user. Read and write permissions for the controlling terminal (`/dev/tty`) are required for this option. If you type a null line, the file is skipped. You should use this option only with the -i or -o options.

-t

Print a table of contents of the input. No files are created. You can use this option only with the -i option.

-u

Copy files unconditionally. Usually an older file doesn't replace a new file with the same name. You can use this option only with the -i or -p options.

-v

Be verbose. Print the names of the affected files. You can use this option only with the -i option. It provides a detailed listing when used with the -t option.

pattern

Simple regular expression given in the name-generating notation of the shell.

directory

The destination directory.

Copyright © 2014, QNX Software Systems Limited

Description:

The `cpio` utility produces and reads files in the format specified by the POSIX `cpio` Archive/Interchange File Format. It operates in three modes.

The -i mode (copy in) extracts files from the standard input, which is assumed to be the product of a previous `cpio -o`. Only files with names that match patterns are selected. Multiple patterns may be specified. If no patterns are specified, the default for patterns is to select all files. The extracted files are conditionally created and copied into the current directory, and possibly any levels below, based on the options used. The permissions of the files are those stored by the previous `cpio -o` invocation. The owner and group of the files are that of the current user unless the user has appropriate privileges, which causes `cpio` to retain the owner and group of the files stored by the previous `cpio -o` invocation.

The -o mode writes the archive to the standard output.

The -p mode (pass) reads the standard input to obtain a list of pathnames of files that are conditionally created and copied into the destination directory based upon the options used.

If an error is detected, the cause is reported and the `cpio` utility continues to copy other files. The utility skips over any unrecognized files encountered in the archive.

The following restrictions apply to the `cpio` utility:

- Pathnames are restricted to 256 characters.
- Appropriate privileges are required to copy special files.
- Blocks are reported in 512-byte quantities.
- Leading slashes (`/`) are stripped when files are extracted from an archive.

Examples:

Copy out the files listed by the *ls* (p. 1139) utility and redirect them to the file `archive`:

```
ls | cpio -o >archive
```

Use the output file `archive` from the `cpio -o` utility, extract those files that match the patterns `memo/al` and `memo/b*`, create the directories below the current directory, and place the files in the appropriate directories:

```
cpio -id "memo/al" "memo/b*" <archive
```

Take the filenames piped to `cpio` from the *find* (p. 750) utility and copy or link those files to another directory named `newdir`, while retaining the modification time:

```
find . -depth -print | cpio -pdlmv newdir
```

Exit status:

Copyright © 2014, QNX Software Systems Limited

0

All input files were copied.

2

The utility encountered errors in copying or accessing files or directories. An error is reported for nonexistent files or directories or for permissions that don't allow the user to access the source or target files.

Caveats:

When `cpio` restores a directory, it matches the permissions of the directory created to those of the original. If that directory lacks write permission, any attempt to copy additional files under that directory fails. To get around this, save the files under a directory first before the directory itself. If *`find`* (p. 750) is used to generate pathnames for `cpio`, the -depth option should be supplied to `find`.

Note also that the controlling terminal (`/dev/tty`) is used to prompt the user for information when the -i or -r options are specified.

Copyright © 2014, QNX Software Systems Limited

cron

Clock server (UNIX)

Syntax:

```
cron [-d crondir] [-s] [-v] &
```

Runs on:

QNX Neutrino

Options:

-d *crondir*

Use the named directory instead of `/var/spool/cron`.

-s

Poll for jobs every minute (to compensate for clock skew).

-v

Turn on verbose mode. Log and diagnostic messages are written to standard error as `cron` operates.

Description:

The `cron` server schedules commands to be run at specified times, without user intervention. This server supports user-specific `cron` entries, and runs continuously. The server must be run in the background.

The `cron` server assumes it has sole use of the `/var/spool/cron` directory. Therefore, you can run only one `cron` server per filesystem containing that directory. You typically run the `cron` server on the network server.

Commands are specified by instructions found in `crontab` files, which you can access via the *`crontab`* (p. 157) utility.

To minimize overhead, `cron` examines the contents of the files in `/var/spool/cron/crontabs` when it first comes up, and then reexamines only those that have been changed via the `crontab` utility.

Copyright © 2014, QNX Software Systems Limited

Files:

Errors cause diagnostic messages to be written to standard error. If you specify -v, log messages are written to the standard error.

The `cron` utility uses data read from the following:

`/var/spool/cron`

Each `cron` command assumes it has exclusive use of this directory.

`/var/spool/cron/cron.allow`

If present, this file lists the only users authorized to have their crontab run. By default, all users are authorized. The `cron.deny` list (below) overrides the setting of the `cron.allow` list.

`/var/spool/cron/cron.deny`

If present, this file lists users who aren't authorized to have their crontab run. This list overrides the list of users authorized (the `cron.allow` file).

`/var/spool/cron/crontabs/*`

The periodic commands to be run are read out of files found in this directory.

Exit status:

The `cron` utility normally runs indefinitely. However, it terminates early if errors are encountered in startup, errors are encountered in reading the `crontabs` files, or if it's terminated by a signal.

0

`cron` was successfully and cleanly terminated by a `SIGTERM` or `SIGPWR` signal.

>0

An error occurred. A diagnostic message will have been written to standard error.

Copyright © 2014, QNX Software Systems Limited

crontab

Schedule periodic background work (POSIX)

Syntax:

```
crontab [-d cron_dir] [-u user] [file]
crontab [-d cron_dir] [-e | -l | -r] [-u user]
```

Runs on:

QNX Neutrino

Options:

-d *cron_dir*

The location of the `crontab` directory.

-e

Edit a user's `crontab` entry. If you don't specify the -u option, `crontab` edits your own entry. It uses `vi` unless the ***EDITOR*** environment variable names another editor.

-l

("el") List the crontab entry of the user. Without the -u option, the invoking user's entry is listed.

-r

Remove a user's `crontab` entry. Without the -u option, the invoking user's entry is removed.

-u *user*

Specify the user whose crontab is to be acted upon. When submitting a crontab, the new crontab entry replaces or creates that user's crontab. When removing (-r) or listing (-l) existing crontabs, this specifies which user's crontab to remove or list. Only `root` may use this option.

file

The pathname of a file that contains specifications for `crontab` entries (see the Description section for the format for these specifications). If no file is specified, `crontab` uses the standard input.

Copyright © 2014, QNX Software Systems Limited

Description:

The `crontab` utility creates or replaces a user's crontab entry. You can input the new crontab entry by specifying a file that contains specifications for crontab entries. If you don't specify a file, the standard input is used.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Users may use `crontab` if their names appear in the `/var/spool/cron/cron.allow` file. If that file doesn't exist, the file `/var/spool/cron/cron.deny` is checked to determine whether the user should be denied access to `crontab`. If neither file exists, only the superuser is allowed to modify crontab entries. If `cron.allow` doesn't exist and `cron.deny` exists and is empty, then global usage is allowed. These permission files consist of one user name per line.

Each command you specify is executed from your home directory using the shell (`/bin/sh`). The *cron* (p. 155) utility supplies a default environment for the shell, defining ***HOME***, ***LOGNAME***, ***SHELL***(=`/bin/sh`), ***PATH*** (=`:/bin:/usr/bin`) and ***TZ***. Users who want to have their `.profile` executed must explicitly do so in their crontab entry.

A `crontab` entry consists of lines of six fields each. The fields are separated by blanks. The first five are integer patterns that specify the following:

- minute (0-59)
- hour (0-23)
- day of the month (1-31)
- month of the year (1-12)
- day of the week (0-6 with 0=Sunday)

Each of these patterns can be an asterisk (meaning all valid values), an element, a list of elements separated by commas, or a range separated by a hyphen (`-`).

An element is either a number or two numbers separated by a hyphen (meaning an inclusive range). You can specify days by two fields (day of the month and day of the week). If both are specified, both are adhered to.

As an example of specifying the two types of days, the following line:

```
0 0 1,15 * 1
```


Copyright © 2014, QNX Software Systems Limited

runs a command at 00:00h on the first and fifteenth of each month, as well as at 00:00h on every Monday. To specify days with only one field, the other field should be set to *. For example:

```
0 0 * * 1
```

runs a command only on Mondays.

The sixth field of a line in a crontab entry is a string that is executed by the command interpreter at the specified times. A percent character in this field — unless escaped by a backslash — is translated to a `newline` character.

Only the first line (up to a % or end-of-line) of the command field is executed by the command interpreter.

Sample crontab entries

Invoke the `calendar` program each day one minute after midnight:

```
1 0 * * * calendar -
```

Display the current time on the system console every 20 minutes:

```
1,21,41 * * * * (echo -n " "; date; echo) > /dev/con1
```

Clean up the UUCP work directories each weekday at 5:30 am:

```
30 5 * * 1-5 /usr/lib/uucp/uuclean
```

Remove any files under `/tmp` that haven't been modified in more than seven days:

```
0 4 * * * find /tmp -mtime +7 -exec rm -f {} \;
```

If you want the output from your commands, redirect it to a file.

Examples:

List your own crontab entry:

```
crontab -l
```

Files:

When an error occurs, a diagnostic message is written to the standard error.

The standard input may be read for the content of `crontab` files when they are being created (none of -e, -l or -r are specified) and no *file* is specified on the command line.

When a `crontab` file is being edited, the editor invoked by `crontab` inherits `crontab`'s standard input, standard output, and standard error.

Copyright © 2014, QNX Software Systems Limited

The following files relative to one of `/var/spool/cron`, or the directory named in the -d option are used by `crontab`:

`cron.allow`

If this file exists, it is read by `crontab` to determine the exclusive list of userids who are allowed to schedule *`cron`* (p. 155) jobs.

`cron.deny`

If `cron.allow` doesn't exist, `crontab` reads this file to determine a list of userids who are *NOT* allowed to schedule *`cron`* (p. 155) jobs.

`crontabs/`*`userid`*

The `crontab` utility may create, edit, list or remove this file.

Exit status:

0

Successful completion.

>0

An error occurred.

Caveats:

If you inadvertently enter the `crontab` utility with no argument, don't try to get out by typing **Ctrl**–**D** (end-of-file), since this would replace your crontab file with an empty file. In this case, you should exit by typing your interrupt character, which is typically **Ctrl**–**C**.

Copyright © 2014, QNX Software Systems Limited

csplit

Split a file based on the context (POSIX)

Syntax:

```
csplit [-ks] [-f prefix] [-n digits] file arg1 [... argn]
```

Runs on:

QNX Neutrino

Options:

-f *prefix*

Use *prefix* instead of xx to name the output files (i.e. name the created files *prefix*00, *prefix*01, ..., *prefixn* instead of xx00, ..., xx*n*).

-k

Leave previously created files intact. By default, csplit removes created files if an error occurs.

-n *digits*

The number of digits to append to the prefix. The default is 2.

-s

Suppress the output of file size messages.

Description:

The csplit utility splits a file into *n* + 1 sections, as prescribed by *arg1*...*argn*, and then displays the size of the output files.

Each argument specifies where a section ends as follows:

***/rexp/*[*relno*]**

Copy lines until the next line matches the regular expression, *rexp*. A positive or negative *relno* specifies a relative line number from the match.

***%rexp%*[*relno*]**

Same as above, except the lines are discarded instead.

nlines

Copyright © 2014, QNX Software Systems Limited

Copy *nlines* lines from the file.

{*num*}

Repeat the preceding argument this number of times.

Examples:

Extract a file that's encoded with *uuencode* (p. 2038):

```
csplit mail '%^begin%' '/^end/+1' '%.*%'
```

Split a file into three parts of 1, 10, and 100 lines:

```
csplit file 1 10 100
```

Exit status:

0

Success.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

ctags

Generate tags files (POSIX)

Syntax:

```
ctags options files...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Append to the `tags` file, instead of overwriting it.

-B

Use ?*regexp*? instead of */regexp/*.

-D*word*

Ignore *word*. This is handy for parameter macro names.

-e

Include `extern` tags.

-F

Use */regexp/* (the default).

-h

Add hints to help *elvis* (p. 672) distinguish between overloaded tags.

-i

Include inline definitions.

-l

("el") Add a `ln` line number hint (implies -h).

-N

Use line numbers instead of */regexp/*.

-p

Copyright © 2014, QNX Software Systems Limited

Write parsing information to *stdout* (for debugging `ctags`).

-r

Write a `refs` file, in addition to `tags`.

-s

Include `static` tags.

-t

Include `typedefs`.

-v

Include variable declarations.

-x

Write a cross-reference table to *stdout* instead of to the `tags` file.

files

The pathnames of the files that are to be scanned for tags.

Description:

The `ctags` utility generates a file called `tags` from a group of C source files.

Each C source file is scanned for `#define` statements and function definitions. The name of the macro or function becomes the name of a tag. For each tag, a line is added to the `tags` file, which contains:

- the name of the tag
- a tab character
- the name of the file containing the tag
- a tab character
- a way to find the particular line within the file

If you don't specify any options, `ctags` uses `-l -i -s -t -v`.

The *elvis* (p. 672), *more* (p. 1309), *vi* (p. 2040), and *less* (p. 1100), utilities can use entries in the `tags` file to locate and display a definition.

Examples:

Generate tags for all the C source and header files in the current directory:

```
ctags *.[ch]
```


Copyright © 2014, QNX Software Systems Limited

Contributing author:

Steve Kirkendall; `ctags` is part of the `elvis` suite.

Copyright © 2014, QNX Software Systems Limited

cut

Cut out selected fields of each line of a file (POSIX)

Syntax:

```
cut -b list [-n] [file...]
cut -c list [file...]
cut -f list [-d delim] [-s] [file...]
```

Runs on:

QNX Neutrino

Options:

-b *list*

Cut out the bytes found in the byte positions specified by *list*.

-c *list*

Cut out the characters found in the character positions specified by *list*. For example, a *list* of `-c 1-64` outputs the first 64 characters of each line.

-d *delim*

Use the delimiter specified by *delim* (default is tab).

-f *list*

Cut out the fields specified by *list*. For example, `-f 2,9` outputs the second and ninth fields. The fields described by *list* are assumed to be separated in the file by a delimiter character (see option -d). Lines without field delimiters are passed through intact, unless -s is specified.

-n

Don't split multi-byte characters. Characters are output only if at least one byte is selected, and, after a prefix of zero or more unselected bytes, the rest of the bytes that form the character are selected.

-s

If -f is specified, suppress lines with no field delimiters.

file

Copyright © 2014, QNX Software Systems Limited

The pathname of a text file, whose contents are used instead of the standard input.

Description:

For every file you name, the `cut` utility cuts out columns or fields from each line, concatenates them, and writes them to the standard output.

If the fields are of a fixed length, you can select them by the byte or character position with option -b or -c. If, however, the fields vary in length from line to line, you can select them with the -f option, provided they're separated by a delimiter character. By default, `cut` assumes the field delimiter character to be tab. You can use the -d option to specify another delimiter.

In options -b, -c, and -f, *list* is a comma-separated list of integers (in increasing order), with an optional dash (`-`) to indicate ranges.

You can use the `cut` utility as a filter; if no files are given, the standard input is used.

Examples:

The following are examples of the *list* argument:

***list* argument:**	**Meaning:**
`1,4,7`	Select the first, fourth, and seventh characters or fields.
`1-3,8`	Equivalent to 1, 2, 3, 8.
`-5,10`	Equivalent to 1, 2, 3, 4, 5, 10.
`3-`	Equivalent to the third through last.

Map userids to names:

```
cut -d: -f1,5 /etc/passwd
```

List filenames and their permissions:

```
ls -l | cut -c57-79,56,56,1-11
```

Exit status:

0

All input files were output successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 5
D

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`devn-asix.so`	*`devnp-asix.so`* (p. 412)
`devn-speedo.so`	*`devnp-speedo.so`* (p. 449)
`devnp-axe.so`	*`devnp-asix.so`* (p. 412)

This chapter describes the utilities, etc. whose names start with “D”.

Copyright © 2014, QNX Software Systems Limited

date

Display or set the date and time (POSIX)

Syntax:

Display the date and time:

```
date [-tuv] [-s seconds] [+format]
```

Set the date and time:

```
date [-uv] [-S seconds] date
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-S *seconds*

Set the maximum number of seconds (of real time) over which `date` is to adjust the time. The `date` utility doesn't increase the clock speed by more than 100% or decrease it by more than 50%. If `date` can't do a slow adjustment within those constraints, the time is changed immediately. (The default is 300 seconds; use -S0 to disable the gradual adjustment.)

-s *seconds*

Display the string equivalent of this date, supplied as seconds since the start of the Epoch (00:00 January 1, 1970). This value is used instead of the system time value for the number of seconds.

-t

Display the current operating system time, in seconds since the start of the Epoch, as a `long` integer.

-u

Perform operations using Coordinated Universal Time (UTC) instead of local time. UTC is the standard term for Greenwich Mean Time (GMT).

-v

Be verbose.

date

Copyright © 2014, QNX Software Systems Limited

A date specification to set the date to. Only the superuser (`root`) can change the date. For more information, see "*Setting the date* (p. 171)," below.

+format

The format in which the date and time are to be displayed.

Description:

The `date` utility is used to display and set the current system date and time in software. Only the superuser (`root`) may use `date` to set the time.

Displaying the date

The `date` utility normally displays the current date and time according to the operating system's internal time, maintained in software as the number of seconds since the Epoch (00:00 January 1, 1970). When the -s *seconds* option is used, `date` uses the value specified by the *seconds* argument instead of the current OS time.

You can specify the format and content of the displayed date and time with the *+format* option. The *format* is composed of ASCII characters and field descriptors prefaced with %, in a manner similar to a C-language *printf()* format specifier (the specific characters used to specify field types are, however, completely different). In the output, each field descriptor is replaced by its corresponding value; all other characters are copied to the output without change.

This utility uses *strftime()*, a `libc` library function, to format the time into a string. For a complete list of the field descriptors you can use in the *+format* option, see *strftime()* in the *C Library Reference*.

The `date` utility always terminates its output with a newline character.

Setting the date

If you're a system administrator running as `root`, you may use `date` to set the system time. To set the hardware clock to match the current system time set by `date`, you should use the *`rtc`* (p. 1714) utility.

Be careful if you set the date during the period that a time zone is switching from daylight saving time to standard time. When a time zone changes to standard time, the local time goes back one hour (for example, 2:00 a.m. becomes 1:00 a.m.). The local time during this hour is ambiguous (e.g. 1:14 a.m. occurs twice in the morning that the time zone switches to DST). To avoid problems, use UTC time to set the date in this period.

By default, if the new time is in the range of:

(-2.5 minutes + old time, 5 minutes + old time)

Copyright © 2014, QNX Software Systems Limited

the `date` utility makes a "slow adjustment" — it increases the clock speed by less than 100% or decreases the clock speed by less than 50% over a period of time from 1 second to 5 minutes until the clock catches up with the new time. This slow adjustment doesn't cause major discontinuities in the time flow. You can disable the slow adjustment by using the -S0 option.

The `date` utility recognizes three formats for setting the time:

1. [[[*CC*]*YY*]*MM*]*DD*]*hhmm*[.*SS*]
2. *MMDDhhmm*[*YY*]
3. *DD* [*Month* [[*CC*]*YY* [*hh* [*mm* [*SS*]]]]] [am|pm]

Where:

CC

Century (e.g. 19 if the year is 1997)

YY

Year modulo 100 (e.g. 97 if the year is 1997)

MM

Numerical month of the year (01 for January, 02 for February, etc.)

Month

Either the numerical month (1, 2,...12) or the standard English abbreviation for the month (jan, feb,...dec)

DD

Day of the month

hh

Hour of the day

mm

Minute of the hour

SS

Seconds of the minute

am|pm

Literally am or pm; you can combine them with hour values less than 13 if you don't want a 24-hour clock

Copyright © 2014, QNX Software Systems Limited

Format 1 is compatible with the *touch* (p. 1968) utility. Since each field is two digits, you must specify a leading 0 for single-digit numbers. You should find this format particularly useful for adjusting the time of day, since its minimal form is just *hhmm* (hour and minute).

Format 2 follows the UNIX System V date conventions. It's similar to the Format 1, with the month and day specified, but the year is optional at the end of the specification instead of the beginning. If there's a dot (`.`) in the date, `date` assumes the date is Format 1 instead of Format 2. The `date` utility also differentiates between *MMDDhhmmYY* (Format 2) and *YYMMDDhhmm* (Format 1) by the value of the first pair of digits. The years 00-12 are before the Epoch. Therefore, if the first pair of digits is in that range, the date is treated as it is in Format 1.

Format 3 follows the date convention used in QNX 4.00 and earlier. This format is assumed if there's more than one operand (the other two formats consist of a single string of digits), or if there's just one number that's two or fewer digits in length.

If you change the date or time, `date` adds a a line to the `/var/log/wtmp` if it already exists.

The `date` utility doesn't create `/var/log/wtmp` if it doesn't already exist. This file can quickly become very big, which isn't good on an embedded system with limited resources.

Examples:

Display the date and time on separate lines:

```
$ date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"
DATE: 01/20/99
TIME: 08:51:59
```

Display the time in 12-hour format:

```
$ date "+TIME: %r"
TIME: 01:36:32 PM
```

Set the system date to 22 February 1997:

```
date 22 2 97
```

Set the system date and time to 16 May 1997, 4:30 pm:

```
date 16 may 1997 4 30 pm
```

Adjust the system time to 4:34 pm; use today's date:

```
date 1634
```

Copyright © 2014, QNX Software Systems Limited

The following command, which illustrates the use of `date -s`, displays the date of the last entry in the `/usr/adm/syslog` file (in this file, the first column of each record is the time in seconds since the Epoch):

```
$ date -s `tail -n1 /usr/adm/syslog | cut -f1 -d ' '`
Wed Apr 15 14:25:49 EDT 1997
```

For more information, see *cut* (p. 166), *logger* (p. 1119), and *tail* (p. 1888).

Files:

/var/log/wtmp

If you change the data or time, and this file already exists, an entry is added to it to log the change.

Environment variables:

TZ

Specifies the local time zone. The value of ***TZ*** affects the conversion between the system clock (UTC) and the local time.

Exit status:

0

The date was displayed or set successfully.

>0

An error occurred.

Caveats:

Some field descriptors are of unspecified format when not in the POSIX locale. As a result, parsing the output of `date` may be difficult in other locales. QNX Neutrino currently supports only the POSIX (i.e. C) locale.

Copyright © 2014, QNX Software Systems Limited

dcheck

Check a disk for bad blocks (QNX)

Syntax:

```
dcheck [options] drive
```

Runs on:

QNX Neutrino

Options:

-B *max_blks*

The maximum number of blocks to read at a time; *max_blks* can be up to 32 (the default).

-b *blk_cnt*

The maximum number of blocks to check.

-f *first_blk*

The first block to check.

-L *loops*

Loop as in the -l option, but with a specified number of loops.

-l

("el") Loop until input (switching serial/random).

-m

In the bitmap, mark bad blocks as unavailable.

-p

Prompt before starting.

-q

Be quiet; don't display progress information.

-r

Use a random head-movement algorithm.

Copyright © 2014, QNX Software Systems Limited

-V

Verify write after read.

-v

Be verbose; display every bad block on the disk.

-w

Write after read (nondestructive) check.

drive

The name of the disk (e.g. `/dev/fd0`, `/dev/hd0t77`) or the root of the filesystem.

Description:

The `dcheck` utility verifies that a disk has been correctly formatted, by attempting to read every block on the drive. The block numbers of any blocks that can't be read are displayed (in hex) to standard output. A summary of the total number of bad blocks is also displayed. You can use `dcheck` to check any formatted disk, including disks that contain files. The files aren't damaged.

If you don't specify the number of blocks to verify, `dcheck` obtains this information from the filesystem and checks *all* the blocks on the specified drive.

If a disk has been initialized for a QNX 4 filesystem, you should use the -m option to remove any bad blocks from the disk-allocation bitmap (`/.bitmap`). This is especially true for hard disks. When you specify the -m option, `dcheck` attempts to read the file `/.bad_blks` from the disk. This file contains a list of all known bad blocks, in sorted order. If `/.bad_blks` is found, `dcheck` reads it, and when it's finished checking the disk, `dcheck` updates the bitmap and recreates the `/.bad_blks` file. Note that the `dcheck` utility only adds to, but never removes, bad block information in this file.

Some blocks may be marginal, so if you run `dcheck` multiple times (see the -l and -L options), you can increase the chance of recognizing these blocks and adding them to the `/.bad_blks` file.

The *chkfsys* (p. 114) utility also recognizes the `/.bad_blks` file.

To help you find any marginal blocks, `dcheck` has a number of options to provide additional checking of a disk. For example, the -r option checks the blocks in a random order; each check consists of a random number of blocks between 1 and 32 (or less, depending on the value specified in the -B option). The `dcheck` utility keeps track of the checked blocks and checks each one only once. This option allows you to find blocks that are bad due to a slight lag time in head movement.

Copyright © 2014, QNX Software Systems Limited

The -I option makes dcheck continuously check the disk until you stop it. For this option, -r is implicitly used and is toggled for each invocation. That is, for the first loop, random checking is set on; for the second loop, it is off, etc. At the end of each complete check, you're prompted to stop the loop. If you don't stop it within 15 seconds, dcheck is started again, etc. The -L option is identical to -I with an upper limit to the number of loops.

The -w option makes dcheck rewrite each block on the device after reading it. This is a nondestructive check that tests the write portion of the hardware. Note that, although this is a more thorough test, it takes more time, depending on the hardware.

The -V option is similar to the -w option, in that dcheck rewrites each block after reading it, but in this case dcheck also rereads each block after the rewrite check and compares this second read with the first. Like the -w option, this test is nondestructive. Note, however, that this is a more thorough test that takes longer.

Examples:

Check all blocks on the hard disk and mark bad blocks in the bitmap:

```
dcheck -m /
```

Check the first 640 blocks on the floppy disk:

```
dcheck -b 640 /dev/fd0
```

Check all blocks on the hard disk:

```
dcheck /dev/hd0t77
```

Files:

If you specify the -m (mark bad blocks) option, the block-special file being checked must be a currently mounted QNX partition. The `.bad_blks` and `.bitmap` files on that filesystem are updated if dcheck discovers any bad blocks.

Exit status:

0

No bad blocks were found.

>0

An error occurred, or bad blocks were found.

Caveats:

The dcheck utility normally opens the disk in read-only mode. However, if you specify the -m, -w, or -V option, the disk is opened in read/write mode. For read/write access,

Copyright © 2014, QNX Software Systems Limited

there must be no open files on the disk, or else dcheck fails with a "Device or resource busy" message. While dcheck is working in read/write mode, no other utilities or programs is allowed to access the disk.

When using the -m option, if dcheck is terminated by a SIGBREAK or other signal, any pending bad blocks may not be recorded. In any event, the results are nondestructive.

Copyright © 2014, QNX Software Systems Limited

dd

Convert a file while copying it (UNIX)

Syntax:

```
dd [if=input_file] [of=output_file] [options]
```

Runs on:

QNX Neutrino

Options:

if=*input_file*

Read from *input_file* instead of the standard input.

of=*output_file*

Write to *output_file* instead of the standard output. Unless conv=notrunc is given, truncate the file to the size specified by seek= (0 bytes if seek= isn't given).

ibs=*bytes*

Read *bytes* bytes at a time.

obs=*bytes*

Write *bytes* bytes at a time.

bs=*bytes*

Read and write *bytes* bytes at a time. Override ibs and obs.

cbs=*bytes*

Convert *bytes* bytes at a time.

skip=*blocks*

Skip *blocks* ibs-sized blocks at start of input.

seek=*blocks*

Skip *blocks* obs-sized blocks at start of output.

count=*blocks*

Copyright © 2014, QNX Software Systems Limited

Copy only *blocks* ibs-sized input blocks.

conv=*conversion*[,*conversion*...]

Convert the file as specified by the *conversion* arguments. Conversions are:

ascii

Convert EBCDIC to ASCII.

ebcdic

Convert ASCII to EBCDIC.

ibm

Convert ASCII to alternate EBCDIC.

block

Pad newline-terminated records to size of cbs, replacing newline with trailing spaces.

unblock

Replace trailing spaces in cbs-sized block with newline.

lcase

Change uppercase characters to lowercase.

ucase

Change lowercase characters to uppercase.

swab

Swap every pair of input bytes. Unlike the UNIX dd, this works when an odd number of bytes are read. If the input file contains an odd number of bytes, the last byte is simply copied (since there's nothing to swap it with).

noerror

Continue after read errors.

notrunc

Don't truncate the output file.

sync

Pad every input block to size of ibs with trailing NULs.

Copyright © 2014, QNX Software Systems Limited

You can follow all numbers by a multiplier:

b

Blocks (×512).

k

Kbytes (×1024).

w

Words (×2).

x*m*

Multiply by *m*.

Description:

The dd utility copies a file (from the standard input to the standard output, by default) with a user-selectable blocksize, while optionally performing conversions on it. It's meant for writing raw data directly to devices such as tape and disk or writing over the network, with control over blocking factors and character set translations.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

You can use this command for copying partial files. You can specify the block size, skip count, and the number of blocks to copy. Sizes are in bytes by default; you can append the letters w, b, or k to the number to indicate words (2 bytes), blocks (512 bytes), or K (1024 bytes). When dd is finished, it reports the number of full and partial blocks read and written.

Examples:

Copy file `file1` to `file2`, converting all text to lowercase letters:

```
dd if=file1 of=file2 conv=lcase
```

Exit status:

>0

An error occurred.

0

Copyright © 2014, QNX Software Systems Limited

The copy and translation operation was successful.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

deflate

Compress files for flash filesystems

Syntax:

```
deflate [options] [filename]...
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-b *size*

The compression block size; one of 4K, 8K, 16K or 32K (default: 8K). The K is assumed; you don't need to specify it.

-o *fname*

The output filename. A filename of - means standard output. By default, `deflate` compresses the files in place.

-i

Inflate files (default: deflate).

-t 1|2

The compression type; the default is 2. For a comparison of the types, see below.

-v

Be verbose; list information on each file as it's compressed.

filename...

The files to compress. If no files are given and the -i option is specified, `deflate` reads from standard input and writes to standard output, allowing it to be used as a filter.

Description:

The `deflate` utility compresses files for a flash filesystem. It's intended to be used in conjunction with the filter attribute for *`mkefs`* (p. 1209). It can also be used to precompress files intended for a flash filesystem.

Copyright © 2014, QNX Software Systems Limited

The compression types (specified with the -t option) are:

Type	Compression Speed	Decompression Speed	Compression Amount
1	Fast	Very fast	30% on executables
2	Slow	Fast	45% on executables

Examples:

Deflate all executables that are to be placed on an embedded target:

```
deflate -v /target/bin/* /target/lib/*
```

Inflate a previously deflated file:

```
deflate -i deflated_file
```

Deflate a file without changing the input file:

```
deflate -o file.dfl file
```


Copyright © 2014, QNX Software Systems Limited

deva-ctrl-4dwave.so

Sound driver for the Trident 4DWave

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d 4dwave [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-4dwave.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-4dwave.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-4dwave.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-4dwave.so` directly from `io-audio`:

```
io-audio -d 4dwave &
```

Mount `deva-ctrl-4dwave.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-4dwave.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-4dwave.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-audiopci.so

Sound driver for the AudioPCI chip family

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d audiopci [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-audiopci.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-audiopci.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-audiopci.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-audiopci.so` directly from `io-audio`:

```
io-audio -d audiopci &
```

Mount `deva-ctrl-audiopci.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-audiopci.so &
```

Files:

`deva-mixer-ac97.so` or **`deva-mixer-ak4531.so`**

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-audiopci.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Contributing author:

This utility is based on software contributed to The NetBSD Foundation by Lennart Augustsson <`augustss@netbsd.org`> and Charles M. Hannum.

License:

This utility is based on software from The NetBSD foundation; for licensing information, see the Third Party License Terms List at *`http://licensing.qnx.com/third-party-terms/`*.

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-cs4281.so

Sound driver for the CS4281

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d cs4281 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-cs4281.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci ***xx***

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-cs4281.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-cs4281.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-cs4281.so` directly from `io-audio`:

```
io-audio -d cs4281 &
```

Mount `deva-ctrl-cs4281.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-cs4281.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-cs4281.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-ess1938.so

Sound driver for the ESS1938

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d ess1938 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-ess1938.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci ***xx***

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-ess1938.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-ess1938.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-ess1938.so` directly from `io-audio`:

```
io-audio -d ess1938 &
```

Mount `deva-ctrl-ess1938.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-ess1938.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-ess1938.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-geode.so

Sound driver for the National Semiconductor Geode family of chips

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d geode [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-geode.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci ***xx***

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-geode.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-geode.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-geode.so` directly from `io-audio`:

```
io-audio -d geode &
```

Mount `deva-ctrl-geode.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-geode.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-geode.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Caveats:

You need a current BIOS to run `deva-ctrl-geode.so`. The BIOS VSA (Virtual System Architecture) technology must be sufficiently up-to-date to enable the native audio programming techniques used by this driver. If the BIOS is too old, `deva-ctrl-geode.so` will exit and *`slogger`* (p. 1807) will contain the exit message.

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-i8x0.so

Sound driver for the Intel 8X0

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d i8x0 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-i8x0.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci ***xx***

The PCI index of the card you want to attach to. If you don't specify this option, the driver attempts to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-i8x0.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-i8x0.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-i8x0.so` directly from `io-audio`:

```
io-audio -d i8x0 &
```

Mount `deva-ctrl-i8x0.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-i8x0.so &
```

Files:

deva-mixer-ac97.so **(p. 213)**

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-i8x0.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-intel_hda.so

Sound driver for the Intel High Definition Audio controllers

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d intel_hda [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-intel_hda.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If you don't specify this option, the driver attempts to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-intel_hda.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-intel_hda.so` is running, you can use applications with sound, and those that control the sound system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-intel_hda.so` directly from `io-audio`:

```
io-audio -d intel_hda &
```

Mount `deva-ctrl-intel_hda.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-intel_hda.so &
```

Files:

deva-mixer-hda.so **(p. 215)**

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-intel_hda.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-nmg6.so

Sound driver for the NeoMagic 6 family of chips

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d nmg6 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-nmg6.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If you don't specify this option, the driver attempts to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-nmg6.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-nmg6.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-nmg6.so` directly from `io-audio`:

```
io-audio -d nmg6 &
```

Mount `deva-ctrl-nmg6.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-nmg6.so &
```

Files:

deva-mixer-ac97.so **(p. 213)**

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-nmg6.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-sb.so

Driver for Sound Blaster 16 and compatible sound cards.

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -dsb ioport=port,irq=req,dma=ch,dma1=ch &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio -oioport=port,irq=req,dma=ch,dma1=ch \
/lib/dll/deva-ctrl-sb.so &
```

Runs on:

QNX Neutrino

Targets:

x86 only (requires ISA bus)

Options:

dma=*ch*

The primary DMA channel to be used. The value of *ch* may be 0, 1, 3, 5, 6, or 7. The default value is 1.

dma1=*ch*

The secondary DMA channel to be used. The value of *ch* may be 0, 1, 3, 5, 6, or 7. The default value is 0.

ioport=*port*

Specifies the base I/O port for Sound Blaster commands. The value of *port* is usually `0x220`, `0x240`, `0x260`, or `0x280`. The default value is `0x220`.

irq=*req*

The interrupt request line specified by *req* cannot be shared. The default value is 10.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Copyright © 2014, QNX Software Systems Limited

Description:

The `deva-ctrl-sb.so` shared object is a DLL for the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-sb.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Errors:

When an error occurs, `deva-ctrl-sb.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Caveats:

Some sound cards listed as being Sound Blaster compatible default to another mode (e.g. some ESS 18xx series (not ESS 1869), OPTi 931, Yamaha OPL3-SA). These cards will not work with this driver.

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-usb.so

Sound driver for USB audio devices

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d usb [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-usb.so &
```

Runs on:

QNX Neutrino

Options:

busno=*bus*

The bus number of the USB controller.

devno=*dev*

The USB address of the device.

did=*did*

The device ID of the device.

iface=*iface*

The audio control interface number.

max_fragsize=*num*

Maximum fragsize (the default is 8 KB).

rx_voices=*num*

The number of RX voices.

tx_voices=*num*

The number of TX voices.

Copyright © 2014, QNX Software Systems Limited

vid=*vid*

The vendor ID of the device.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-usb.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-usb.so` is running, you can use applications with sound, and those that control the sound-system.

If you're starting the driver to target a particular audio function, you must provide the *vid*, *did*, *busno*, *devno*, and *iface* options. The USB device must be inserted before you start or mount the driver.

When the device is removed, the audio DLL is unmounted from `io-audio`, but the `io-audio` manager itself continues to run (after you re-insert the device, you can run `mount -Tio-audio deva-ctrl-usb.so`). You have to slay `io-audio` if you want it to be terminated.

- Some USB Host controllers are restricted in the number of transfer descriptors available, so adjustments to either the audio driver's max_fragsize or the USB Host controller driver's TD pools (num_td=*xx*) may be required for successful isoch transfers.
- Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Examples:

Invoke `deva-ctrl-usb.so` directly from `io-audio`:

```
io-audio -d usb &
```

Mount `deva-ctrl-usb.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-usb.so &
```

Errors:

When an error occurs, `deva-ctrl-usb.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-via686.so

Sound driver for the VIA686

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d via686 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-via686.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci ***xx***

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-via686.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-via686.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-via686.so` directly from `io-audio`:

```
io-audio -d via686 &
```

Mount `deva-ctrl-via686.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-via686.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-via686.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-via8233.so

Sound driver for the VIA 8233 Audio controller

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d via8233 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-via8233.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

did=*did*

The device ID of the device.

pci *xx*

The PCI index of the card you want to attach to. If you don't specify this option, the driver tries to find the first unused card in the system.

vid=*vid*

The vendor ID of the device.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-via8233.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-via8233.so` is running, you can use applications with sound, and those that control the sound-system.

Copyright © 2014, QNX Software Systems Limited

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Examples:

Invoke `deva-ctrl-via8233.so` directly from `io-audio`:

```
io-audio -d via8233 &
```

Mount `deva-ctrl-via8233.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-via8233.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-via8233.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-vortex.so

Sound driver for the Aureal Vortex

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d vortex [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-vortex.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-vortex.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-vortex.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-vortex.so` directly from `io-audio`:

```
io-audio -d vortex &
```

Mount `deva-ctrl-vortex.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-vortex.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-vortex.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Contributing author:

Aureal Semiconductor Inc.

Copyright © 2014, QNX Software Systems Limited

deva-ctrl-ymfds1.so

Sound driver for the Yamaha DS1

You must be `root` to start this driver.

Syntax:

Direct invocation (also causes a new `io-audio` process to start):

```
io-audio -d ymfds1 [opt[,opt...]] &
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-ymfds1.so &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

pci *xx*

The PCI index of the card you want to attach to. If this option is not specified, the driver will attempt to find the first unused card in the system.

For card options that apply to all sound drivers, see the entry for *io-audio* (p. 989).

Description:

The `deva-ctrl-ymfds1.so` shared object is a device driver DLL used by the *io-audio* (p. 989) manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-ymfds1.so` is running, you can use applications with sound, and those that control the sound-system.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Copyright © 2014, QNX Software Systems Limited

Examples:

Invoke `deva-ctrl-ymfds1.so` directly from `io-audio`:

```
io-audio -d ymfds1 &
```

Mount `deva-ctrl-ymfds1.so` (`io-audio` must be running):

```
mount -Tio-audio /lib/dll/deva-ctrl-ymfds1.so &
```

Files:

`deva-mixer-ac97.so`

Supports the mixer.

Errors:

When an error occurs, `deva-ctrl-ymfds1.so` sends a description of the error to the system logger (see *`slogger`* (p. 1807)).

Contributing author:

Aureal Semiconductor Inc.

Copyright © 2014, QNX Software Systems Limited

deva-mixer-ac97.so

Mixer DLL for the Intel Audio Codec '97 (AC'97)

You can't invoke this shared object. An audio driver (`deva-ctrl-*`) loads it if the driver determines that your hardware needs it.

Syntax:

N/A

Runs on:

QNX Neutrino

Options:

None.

Description:

The `deva-mixer-ac97.so` shared object provides an interface between the AC'97 standard codec and an audio driver. For information about the Audio Codec '97 specification, see ihe Intel Corporation website.

Errors:

When an error occurs, `deva-mixer-ac97.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-mixer-ak4531.so

Mixer DLL for the ASAHI KASEI AK4531 CODEC

You can't invoke this shared object. An audio driver (`deva-ctrl-*`) loads it if the driver determines that your hardware needs it.

Syntax:

N/A

Runs on:

QNX Neutrino

Options:

None.

Description:

The `deva-mixer-ak4531.so` shared object provides an interface between the AK4531 CODEC and an audio driver.

Errors:

When an error occurs, `deva-mixer-ak4531.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-mixer-hda.so

Mixer DLL for High Definition Audio codecs

You can't invoke this shared object. An audio driver (`deva-ctrl-*`) loads it if the driver determines that your hardware needs it.

Syntax:

N/A

Runs on:

QNX Neutrino

Targets:

x86

Options:

None.

Description:

The `deva-mixer-hda.so` shared object provides an interface between High Definition Audio codecs and an audio driver.

The DII has support for a subset of the codecs in production; your particular codec may not yet be supported.

Errors:

When an error occurs, `deva-mixer-hda.so` sends a description of the error to the system logger (see *slogger* (p. 1807)).

Copyright © 2014, QNX Software Systems Limited

deva-util-restore.so

Shared object used to restore an audio driver's state

You don't invoke this object directly; *io-audio* (p. 989) loads it on loading a new driver.

Syntax:

N/A

Runs on:

QNX Neutrino

Targets:

ARM, x86

Options:

None.

Description:

When `io-audio` loads a new driver, it uses the `deva-util-restore.so` shared object to restore the driver's state, which consists of all the device settings, such as the output volume level and the selected input connection. The state information is restored from a configuration file that was updated the last time the driver ran.

This shared object is an optional component of the audio system. If `io-audio` can't load it, `io-audio` continues, but doesn't restore the driver state. See also the `io-audio` global option config_write_delay.

Errors:

When an error occurs, `deva-util-restore.so` sends a description of it to the system logger, *slogger* (p. 1807).

Copyright © 2014, QNX Software Systems Limited

devb-adpu320

Driver for Adaptec AIC-7901/7902-based SCSI adapters (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-adpu320 [cam option[,option]...]
          [cdrom option[,option]...]
          [disk option[,option]...]
          [optical option[,option]...]
          [adpu320 option[,option]...]
          [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas (`,`) to separate the options. You can put the cam, cdrom, disk, optical, adpu320, and blk groups of options in any order.

cam options

The cam options control the common access methods:

lun=*mask*

Enable Logical Unit Number (LUN) scan for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is `0x00`.

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose: display full information about SCSI units (devices) on startup.

Copyright © 2014, QNX Software Systems Limited

cdrom options

The cdrom options control the driver's interface to *cam-cdrom.so* (p. 89). If specified, they must follow the cdrom keyword.

disk options

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

optical options

The optical options control the driver's interface to *cam-optical.so* (p. 93). If specified, they must follow the optical keyword.

adpu320 options

The adpu320 options control the driver's interface to the U320 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the adpu320 keyword before each controller's set of options.

vid= ***vid***

The vendor ID of the controller.

did=*did*

The device ID of the controller.

pci=*index*

The PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Description:

The `devb-adpu320` driver is for SCSI adapters based on the Adaptec AIC-790X chips.

Controllers supported by this driver include, but aren't necessarily limited to:

Manufacturer	Controller
Adaptec	AIC-7901
Adaptec	AIC-7902

Copyright © 2014, QNX Software Systems Limited

Manufacturer	Controller
Adaptec	29320A-R

If you have problems with the PCI adapter make sure that you have an up to date version of the the adapter BIOS as well as system BIOS.

Controllers are numbered from 0 to n, in the order they're found. For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately.

The `devb-adpu320` driver closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and are written to standard error.

Examples:

Assume an U320 controller, and list all connected devices:

```
devb-adpu320 &
```

Assume an U320 PCI controller with a PCI index of 1, and list all connected devices:

```
devb-adpu320 adpu320 pci=1 &
```

Files:

The `devb-adpu320` driver causes *`io-blk.so`* (p. 993) to adopt various block special devices under `/dev`. These devices are normally named `hd` *n* (or `cd` *n* for CD-ROMs), where *n* is the physical unit number of the device. This driver could also require the following shared objects:

Binary	Required
`cam-cdrom.so` (p. 89)	For CD-ROM access
`cam-disk.so` (p. 91)	For hard-disk access
`libcam.so`	Always

Exit status:

The `devb-adpu320` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

Copyright © 2014, QNX Software Systems Limited

The `devb-adpu320` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
`/dev/hd0t77`	`/hd`	`qnx4`
`/dev/cd0`	`/cd`	`cd`
`/dev/hd0t6`	`/dos`	`dos`
`/dev/hd0t11`	`/dos`	`dos`

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-aha8

Driver for Adaptec AIC-7870/7880 based SCSI adapters (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-aha8 [cam option[,option]...]
          [cdrom option[,option]...]
          [disk option[,option]...]
          [optical option[,option]...]
          [aha8 option[,option]...]
          [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas (`,`) to separate the options. You can put the cam, cdrom, disk, optical, aha8, and blk groups of options in any order.

cam options

The cam options control the common access methods:

lun=*mask*

Enable Logical Unit Number (LUN) scan for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is `0x00`.

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose: display full information about SCSI units (devices) on startup.

Copyright © 2014, QNX Software Systems Limited

cdrom options

The cdrom options control the driver's interface to *cam-cdrom.so* (p. 89). If specified, they must follow the cdrom keyword.

disk options

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

optical options

The optical options control the driver's interface to *cam-optical.so* (p. 93). If specified, they must follow the optical keyword.

aha8 options

The aha8 options control the drivers interface to the AHA 8 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the aha8 keyword before each controller's set of options.

pci=*index*

The PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

noreset

Don't reset the SCSI bus.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Description:

The devb-aha8 driver is for SCSI adapters based on the Adaptec AIC-7870 and AIC-7880 chips. Controllers supported by this driver include, but aren't necessarily limited to:

Manufacturer	Controller
Adaptec	AIC-7870
Adaptec	AIC-7880
Adaptec	AHA-2940
Adaptec	AHA-2940W

Copyright © 2014, QNX Software Systems Limited

Manufacturer	Controller
Adaptec	AHA-3940

The devb-aha8 driver queries the BIOS for PCI card memory addresses.

If you have problems with the PCI adapter make sure that you have an up to date version of the the adapter BIOS as well as system BIOS.

Controllers are numbered from 0 to *n*, in the order they're found.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately.

The devb-aha8 driver closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and are written to standard error.

Examples:

Assume an AHA 8 controller, and list all connected devices:

```
devb-aha8 &
```

Assume an AHA 8 PCI controller with a PCI index of 1, and list all connected devices:

```
devb-aha8 aha8 pci=1 &
```

Files:

The devb-aha8 driver causes *io-blk.so* (p. 993) to adopt various block special devices under /dev. These devices are normally named hd*n* (or cd*n* for CD-ROMs), where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
cam-cdrom.so (p. 89)	For CD-ROM access.
cam-disk.so (p. 91)	For hard-disk access.
libcam.so	Always

Exit status:

The devb-aha8 driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

Copyright © 2014, QNX Software Systems Limited

0

The `devb-aha8` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
`/dev/hd0t77`	`/hd`	`qnx4`
`/dev/cd0`	`/cd`	`cd`
`/dev/hd0t6`	`/dos`	`dos`
`/dev/hd0t11`	`/dos`	`dos`

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-ahci

Driver for AHCI SATA interfaces (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-ahci [cam option[,option]...]
          [mem option[,option]...]
          [ahci option[,option]...]
          [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Options:

Use commas (`,`) to separate the options. You can put the cam, mem, ahci, and blk groups of options in any order.

cam options

The cam options control the common access methods:

lun=*mask*

Enable Logical Unit Number (LUN) scan for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is `0x00`.

quiet

Be quiet: don't display any information on startup.

resmgr=*m*:*l*:*h*:*d*

Enable the `/dev/cam`*X*`/`*XXX* interface and set the maximum (*m*), low (*l*), high (*h*), and devno (*d*) thread-pool parameters. The default is `5:1:2:-1`.

verbose

Be verbose: display full information about SCSI units (devices) on startup.

Copyright © 2014, QNX Software Systems Limited

mem options

name=*tname*

The typed memory name to use.

ahci options

The ahci options control the driver's interface to the AHCI controller. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the ahci keyword before each controller's set of options.

- Interface-specific options:

  did=*did*

  The device ID of the controller.

  ioport=*addr*

  The address of the interface.

  irq=*req*

  The interrupt used by the controller.

  nobmstr

  Don't use busmastering.

  nports=*num*

  Set the number of ports.

  pci=*index*

  The PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

  pi=*bitmap*

  Set the ports implemented bitmap. For example, pi=0x4 specifies port 2.

  port=*N*,*device*

  Specify options for *device* on port *N*.

  priority=*prio*

  Set the priority of the processing thread. The default is 21.

Copyright © 2014, QNX Software Systems Limited

timeout=*timeout*

Set the I/O request timeout, in seconds. The default is 10.

vid=*vid*

The vendor ID of the controller.

xlat=*xlat*

The busmaster translation.

- Device-specific options:

chs

Use Cylinder-Head-Sector mode. The default is LBA.

geometry=*heads:cyl:sect*

Specify the drive geometry.

multiblk=*blks*

Set multiblock mode, using the given number of blocks per interrupt.

nobmstr

Don't use busmastering.

nonremovable

Report the device as being nonremovable.

rahead=*state*

Enable or disable device read-ahead, where *state* is on or off.

smart

Enable SMART monitoring.

verbose=*level*

Set the device verbosity level.

wcache=*state*

Enable or disable device write cache, where *state* is on or off.

blk options

Copyright © 2014, QNX Software Systems Limited

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Description:

The `devb-ahci` supports the Intel AHCI SATA controller with the following device IDs:

- ICH-6 82801FB_SATA 0x2651
- ICH-6 82801FBM_SATA 0x2653
- ICH-7 82801GB_SATA 0x27c1
- ICH-7 82801GBM_SATA 0x27c5

You need to enable AHCI mode in the BIOS.

Examples:

Detect all SATA controllers, and list all connected devices:

```
devb-ahci &
```

Detect all SATA controllers and use `dma` typed memory:

```
devb-ahci mem name=/ram/dma &
```

Files:

The `devb-ahci` driver causes *io-blk.so* (p. 993) to adopt various block special devices under `/dev`. These devices are normally named `hd`*n*, where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
cam-disk.so (p. 91)	For hard-disk access.
`libcam.so`	Always

Exit status:

The `devb-ahci` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

Copyright © 2014, QNX Software Systems Limited

The `devb-ahci` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
`/dev/hd0t77`	`/hd`	`qnx4`
`/dev/hd0t6`	`/dos`	`dos`
`/dev/hd0t11`	`/dos`	`dos`

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-btmm

Driver for BusLogic/Mylex Multimaster host adapters (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-btmm [cam option[,option]...]
          [cdrom  option[,option]...]
          [disk option[,option]...]
          [optical option[,option]...]
          [btmm option[,option]...]
          [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas (`,`) to separate the options. You can put the cam, cdrom, disk, optical, btmm, and blk groups of options in any order.

cam options

The cam options control the common access methods:

lun=*mask*

Enable Logical Unit Number (LUN) scanning for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is `0x00`.

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose: display full information about SCSI units (devices) on startup.

Copyright © 2014, QNX Software Systems Limited

cdrom options

The cdrom options control the driver's interface to *cam-cdrom.so* (p. 89). If specified, they must follow the cdrom keyword.

disk options

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

optical options

The optical options control the driver's interface to *cam-optical.so* (p. 93). If specified, they must follow the optical keyword.

btmm options

The btmm options control the drivers interface to the BusLogic/Mylex Multimaster series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the btmm keyword before each controller's set of options.

ioport=*port*

The I/O port of the interface. By default, it's detected automatically.

clone

Use this for clones; the default is for the driver to attempt to verify the type of card, and clone disables this check.

dma=*channel*

Use the specified DMA *channel*.

irq=*req*

Assume that the controller is using this interrupt. Default is 11.

noreset

Reset the controller and SCSI bus at initialization time.

scsiid=*id*

Specify the SCSI ID used by the controller. Default is 7.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devb-btmm` driver is for the BusLogic/Mylex Multimaster and compatible SCSI controllers.

Controllers supported by this driver include, but aren't necessarily limited to:

Controller	Description
BT-440C	Bus master VL SCSI host adapter.
BT-445C	Bus master VL SCSI host adapter with floppy controller.
BT-542B	Bus master ISA SCSI host adapter with floppy controller.
BT-542D	Bus master ISA-to-Fast SCSI host adapter with floppy controller.
BT-545C	Bus master ISA SCSI host adapter with floppy controller.
BT-545S	Bus master ISA-to-Fast SCSI host adapter with floppy controller.
BT-646S	Bus master MCA SCSI host adapter.
BT-747S	Bus master EISA SCSI host adapter.
BT-946C	Bus master PCI-to-Fast SCSI host adapter with floppy controller.

The driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately.

The `devb-btmm` driver closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and are written to standard error.

Examples:

Start the Multimaster driver:

```
devb-btmm &
```

Start the Multimaster driver with an I/O port of `0x330` and an interrupt of `11`:

```
devb-btmm btmm ioport=0x330,irq=11 &
```


Copyright © 2014, QNX Software Systems Limited

Files:

The devb-btmm driver causes *io-blk.so* (p. 993) to adopt various block special devices under /dev. These devices are normally named hd*n* (or cd*n* for CD-ROMs), where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
cam-cdrom.so (p. 89)	For CD-ROM access
cam-disk.so (p. 91)	For hard-disk access
libcam.so	Always

Exit status:

The devb-btmm driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The devb-btmm driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

The BusLogic/Mylex Multimaster host adapter is compatible with the Adaptec AIC-154x SCSI controller. On startup, the enumerators recognize this adapter as the Adaptec card but devb-aha4 will fail unless it includes the clone option.

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
/dev/hd0t77	/hd	qnx4
/dev/cd0	/cd	cd
/dev/hd0t6	/dos	dos
/dev/hd0t11	/dos	dos

Copyright © 2014, QNX Software Systems Limited

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()* as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-eide

Driver for ATA/IDE disk interface and ATAPI CD-ROM interface (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-eide [blk option[,option]...]
          [cam option[,option]...]
          [cdrom option[,option]...]
          [disk option[,option]...]
          [eide option[,option]...] &
```

Runs on:

QNX Neutrino

Options:

Use commas (`,`) to separate the options. You can put the blk, cam, cdrom, disk, and eide groups of options in any order.

blk ***options***

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

cam ***options***

The cam options control the common access methods:

quiet

Be quiet: don't display any information on startup.

resmgr=*m*:*l*:*h*:*d*

Enable the `/dev/cam`*X*`/`*XXX* interface and set the maximum (*m*), low (*l*), high (*h*), and devno (*d*) thread-pool parameters. The default is `5:1:2:-1`.

verbose

Be verbose.

Copyright © 2014, QNX Software Systems Limited

cdrom *options*

The cdrom options control the driver's interface to *cam-cdrom.so* (p. 89). If specified, they must follow the cdrom keyword.

disk *options*

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

eide options

The eide options control the driver's interface to the EIDE controller. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the eide keyword before each controller's set of options.

- Interface-specific options:

  altstatus

  Use alternate status register for polling. Off by default.

  bs=*board_specific*

  Board-specific options.

  chnl=*chnl*

  The channel number of the controller (0 or 1).

  decode=*xor*

  Set the layout between I/O registers. The default is 0.

  did=*did*

  The device ID of the controller.

  enable

  Enable the chipset interface.

  ioport=*pri***[:***sec***]**

  The I/O port of the interface. By default, it's detected automatically. Use the vaddr option if this is a virtual address.

  irq=*req*

  The interrupt used by the controller.

  iwaitnbsy=*ms*

Copyright © 2014, QNX Software Systems Limited

The amount of time, in milliseconds, to wait for a not-BSY status after interrupt. The default is 20 ms.

master=*device*

Specify master device options. For device-specific options, see below.

nobios

Don't use BIOS transfer mode settings. The default is to use them.

nobmstr

Don't use busmastering. Specify this option if you want to disable DMA.

nodefect

Don't match the device defect list.

nolegacy

Don't scan legacy addresses (`0x1f0`, `0x170`).

nomaster

Don't scan for master devices.

noreset

Don't reset devices at initialization.

noslave

Don't scan for slave devices.

pci=*index*

The PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

priority=*prio*

Set the priority of the processing thread. The default is 21.

resets=*num*

The number of times to retry initialization resets. The default is 1.

slave=*device*.

Copyright © 2014, QNX Software Systems Limited

Specify slave device options. For device-specific options, see below.

stride=*space*

Set the spacing offset between I/O ports (IDE command registers). For example, if the ports are located on 4-byte boundaries, set *space* to 4. The default is 1.

timeout=*timeout*

Set the I/O request timeout in seconds. The default is 10.

tmem=*name*

Set the shared memory region. The default is 0.

vaddr

The port specified by the ioport is a virtual address. By default, it's a physical address.

verbose=*level*

Set the EIDE verbosity level.

vid=*vid*

The vendor ID of the controller.

- Device-specific options:

apm_level=*level*

Set the APM level (`0x7f`–`0xfe`). The default is the maximum (`0xfe`).

ata

Set the device type to ATA.

atapi

Set the device type to ATAPI.

chs

Use Cylinder-Head-Sector mode instead of Logical Block Addressing. LBA is used by default.

drdy=*mode*

Copyright © 2014, QNX Software Systems Limited

Set the read/write Device Ready (DRDY) mode (drdy=off to disable it, or drdy=on to enable).

geometry=*heads*:*cyl*:*sect*

Specify the drive geometry.

mdma=*mode*

Set multi-word DMA mode. Values for *mode* can be 0-2 (or `off` to disable).

multiblk=*blks*

Set the number of blocks per interrupt for multiblk mode.

nobmstr

Don't use busmastering.

nonremovable

Report the device as nonremovable.

pio=*mode*

Set PIO mode. Values for *mode* can be 0-4 (or `off` to disable PIO).

rahead=*state*

Enable or disable the device read cache (*state* is `on` or `off`).

smart

Enable SMART monitoring.

spinup=*time*

The length of time, in seconds, to wait for the device to become ready.

You must also specify the device type (e.g., master=ata).

udma=*mode*

Set ultra DMA mode. Values for *mode* can be 0-6 (or `off` to disable).

verbose=*level*

Copyright © 2014, QNX Software Systems Limited

Set the device verbosity level.

wcache=on | off

Enable or disable the device write cache.

xfer=*width*

Set the I/O access width (8, 16, or 32 bits).

Description:

The `devb-eide` driver is for the IDE (Integrated Drive Electronics), EIDE (Enhanced IDE), and ATA (AT Attachment) hard disk interfaces, as well as the ATAPI (ATA Packet Interface) CD-ROM interface. This driver autodetects all interfaces.

If you're installing multiple operating systems on the drive, make sure they all use a compatible mode. For example, if your drive is 528 MB and DOS will also be installed on the drive, the driver should be configured to use LBA.

The `devb-eide` driver's order of preference for the connection modes is as follows:

1. UDMA
2. MDMA
3. SDMA
4. PIO

If the underlying hardware supports a mode, it's automatically enabled, and the driver selects the best available mode. If you want the driver to use a lower mode, you need to explicitly disable the higher, better modes. For example, if you want the driver to use PIO, and the hardware also supports UDMA and MDMA, you need to explicitly disable UDMA and MDMA.

The `devb-eide` driver uses DMA by default. If you want to disable DMA, specify the nobmstr command-line option.

By default, the driver uses LBA (Logical Block Addressing) modes if the drive supports them. If you want the device programmed to CHS (Cylinder-Head-Sector) mode, specify the chs option.

The `devb-eide` driver closes its standard input, standard output, and standard error immediately after completing its initializations. Any error messages produced during the initialization phase are written to standard error.

Copyright © 2014, QNX Software Systems Limited

Examples:

Detect all IDE controllers, and list all connected devices:

```
devb-eide &
```

Detect an IDE controller at a specific I/O port address and IRQ number, and list all connected devices:

```
devb-eide eide ioport=0x1f0,irq=14
```

Detect a PCMCIA disk that is configured in contiguous I/O mapped addressing at a specific I/O port address and IRQ number:

```
devb-eide eide ioport=0x320:0x32c,irq=7,noslave
```

For PCMCIA devices configured in contiguous I/O mapped addressing, you should always specify the control block address of the interface by adding an offset (usually `12`) to the base address of the port. This isn't required for legacy addressing (`0x1f0` or `0x170`), where the driver adds the standard control block offset (`0x200`) automatically.

Detect an IDE controller with specific vendor and device identifiers, and list all connected devices:

```
devb-eide eide vid=0x8086,did=0x2411,pci=0,chnl=0
```

Detect an IDE controller with a specific vendor ID, device ID, and channel number, and disable ultra DMA on the master:

```
devb-eide eide vid=0x8086,did=0x2411,pci=0,chnl=1,master=udma=off
```

Pass cache and delwri options to `io-blk.so`, uid and gid options to `fs-cd.so`, and vollabel option to `fs-dos.so`:

```
devb-eide blk cache=2m,delwri=2s cd uid=234,gid=120 dos \
vollabel=ignore &
```

The cd and dos options apply to any filesystems of those types that are mounted (either by the automatic mounter or a later explicit mount).

You can also pass generic mount options (as described in *io-blk.so* (p. 993)) as follows:

```
devb-eide blk noatime dos hidden=show,noexec qnx4 ro &
```

This sets the ST_NOATIME mount bit for all filesystems, the ST_RDONLY bit for any QNX 4 filesystem, and the ST_NOEXEC bit for any DOS filesystem. The mount message also has these bits, which apply only to that mountpoint.

Copyright © 2014, QNX Software Systems Limited

Files:

The `devb-eide` driver causes *io-blk.so* (p. 993) to adopt various block special devices under `/dev`. These devices are normally named `hd`*n* (or `cd`*n* for CD-ROMs), where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
cam-cdrom.so (p. 89)	For CD-ROM access
cam-disk.so (p. 91)	For hard-disk access
`libcam.so`	Always

Exit status:

The `devb-eide` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The `devb-eide` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

>0

An error occurred during startup.

Caveats:

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
`/dev/hd0t77`	`/hd`	`qnx4`
`/dev/cd0`	`/cd`	`cd`
`/dev/hd0t6`	`/dos`	`dos`
`/dev/hd0t11`	`/dos`	`dos`

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Copyright © 2014, QNX Software Systems Limited

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-fdc

Driver for floppy disk interface (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-fdc [cam option[,option]...]
         [disk option[,option]...]
         [fdc option[,option]...]
         [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas (`,`) to separate the options. You can put the cam, disk, fdc, and blk groups of options in any order.

cam options

The cam options control the common access methods:

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose.

disk options

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

fdc options

The fdc options control the driver's interface to the fdc controller. If you've installed multiple controllers, you can repeat these options for each controller.

Copyright © 2014, QNX Software Systems Limited

Remember, however, to specify the fdc keyword before each controller's set of options.

ioport=*port*

The I/O port of the interface. The default is `0x3f0`.

irq=*req*

The interrupt used by the controller. The default is 6.

dma=*channel*

Use the specified DMA channel. The default is 2.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Description:

The `devb-fdc` driver is for the floppy disk interface. It autodetects interfaces at address `0x3f0`, interrupt 6, dma channel 2 by default. If you have an interface at a different address/interrupt/dma, specify them to the driver using the fdc options.

The default cache size specified by `io-blk.so` is excessive for `devb-fdc`. You'll probably want to reduce it to something more reasonable:

```
devb-fdc blk cache=512k &
```

Examples:

Assume an FDC interface, and list all connected devices:

```
devb-fdc &
```

Mount a floppy drive that can access QNX or DOS floppy disks:

```
devb-fdc blk cache=128k &
mount -tqnx4 /dev/fd0 /qnxflop
mount -tdos /dev/fd0 /dosflop
```

or:

```
devb-fdc blk cache=128k,automount=+fd0:/qnxflop:qnx4,\
automount=+fd0:/dosflop:dos &
```

Copyright © 2014, QNX Software Systems Limited

Files:

The `devb-fdc` driver causes `io-blk.so` to adopt various block special devices under `/dev`. These devices are normally named `fd`*n*, where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
`cam-disk.so` (p. 91)	For floppy-disk access.
`libcam.so`	Always

Exit status:

The `devb-fdc` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The `devb-fdc` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *`pipe`* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-loopback

Pseudo block driver

In order to start this driver, you must be logged in as `root`.

Syntax:

```
devb-loopback [loopback loopback_option[,option]]
              [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Options:

Use commas (`,`) to separate the options. You can put the blk and loopback groups of options in any order.

loopback options

The loopback options include the following:

The rw, ro, sync, async, denyno, denyrd, denywr, denyrw, blksz=, heads=, and tracks= options apply to all subsequent fd= files, but can be specified multiple times with different values (e.g., loopback ro,fd=/dev/cd0,rw,fd=/fs/hd/iso.img). Multiple fd= files will become `/dev/lo0`, `/dev/lo1`, ... in order.

fd=*path*

Specify a filename (a path registered by your resource manager) to be presented as a block device by `devb-loopback`. This pathname must support *open()*, *close()*, *read()*, *write()*, and *fstat()*; for more information, see "*Driver support* (p. 251)," below.

Each pathname will result in the creation of a corresponding block-special device (see prefix= below), the content of which will be mirrored over the file descriptor using standard read/write calls. You can then mount traditional block filesystems (such as

Copyright © 2014, QNX Software Systems Limited

fs-dos.so (p. 795) and *fs-qnx4.so* (p. 820)) on top of these pseudo devices and use them transparently.

You can specify multiple fd= pathnames; they're managed by a single `devb-loopback` process.

prefix=*name*

The name by which the loopback pseudo block devices are registered. The default is `lo`; thus each fd= entry named on the command line will appear as `/dev/lo0`, `/dev/lo1`, and so on. This is a global option, and applies to all fd= instances.

rw, ro

Set the reading mode with which to open the underlying file descriptor (corresponds to `O_RDWR` or `O_RDONLY`). If you specify read-only, then in turn you can mount only read-only filesystems onto the corresponding `devb-loopback` device (the pseudo-device is advertised up as `DEV_RDONLY`). The default is rw. This option applies to all subsequent instances of fd=.

sync, async

Set the synchronous write mode with which to open the underlying file descriptor (corresponds to `O_SYNC`). The effect of this depends on the resource manager responsible for each file (typically it's ignored).

This option affects writes that are presented to the pseudo-device, and not the write behavior of any mounted filesystems above this (that can be controlled via *mount* (p. 1312) options or blk delwri=... commit=...).

The default is sync. This option applies to all subsequent instances of fd=.

denyno, denyrd, denywr, denyrw

The `SH_DENY` mode with which to open the underlying file descriptor. A deny mode stops other processes from opening the host file, thus preventing the content of it from being changed behind the back of the `devb-loopback` filesystems, which will prevent coherence issues. The default is denyrw. This option applies to all subsequent instances of fd=.

seek, xtype

Copyright © 2014, QNX Software Systems Limited

Configure I/O onto the file descriptor using either an LSEEK+READ combined message or via a READ+XTYPE_OFFSET composite. By default, `devb-loopback` probes the host filesystem as to whether it supports the *pread()* and *pwrite()* API. The default is xtype. This option applies to all subsequent instances of fd=.

blksz=*size*

Override the device blocksize. This is typically probed from a *stat()* or `DCMD_CAM_DEVINFO` *devctl()* call onto the resource manager responsible for each file and the same value advertised up, but some resource managers don't advertise a useful blocksize (in particular, `devf-*` reports 1-byte blocks).

You might also need this option in some cases where a filesystem image was originally formatted on a device with a different sector size to that of the hosting filesystem; it may be necessary for correct operation or performance reasons to mimic the original sector size.

This option applies to all subsequent instances of fd=, but you can reset it to use the probed value with an empty option (e.g. `blksz=2048,fd=/F1,blksz=,fd=/F2`).

nio=*num*

The number of asynchronous I/O threads (which are responsible for executing an internal block device read/write as a normal read/write over the external file descriptor). This multi-threading is useful if you've specified multiple fd= devices.

The default is 1; you can specify a value of 0 to force all I/O to be performed synchronously in the context of the filesystem caller; it can be higher to prevent this I/O from becoming a bottleneck to the filesystem layers above (appropriate only when multiple fd= paths have been specified). This is a global option; the pool of threads services requests to all fd= files in the order of client priority.

heads=*num*

Specify the number of heads (default 1). This option applies to all subsequent instances of fd=.

tracks=*num*

Copyright © 2014, QNX Software Systems Limited

Specify the number of sectors per track (default 1). This option applies to all subsequent instances of fd=.

blk options

The blk options are as for *io-blk.so* (p. 993). If specified, they must follow the blk keyword. For more information, see `io-blk.so`.

Since `devb-loopback` loads the standard block device DLLs, which will create a buffer cache using the same default rules as for the disk filesystems, the cache is likely to be too big for `devb-loopback`. You can reduce the cache size using blk cache=*size*. Depending on the actual size of the device, and the level of caching already implemented by its driver, a value of `128k` may be sufficient.

Description:

The block filesystems in the QNX Neutrino RTOS are implemented as a series of DLL modules and a set of internal APIs. Thus it is possible to mount disk-based filesystems (such as QNX or DOS) only onto released QNX `devb-*` drivers. The `devb-loopback` driver implements a mapping between an arbitrary file descriptor and the block API, allowing any resource manager to be used to host a disk filesystem.

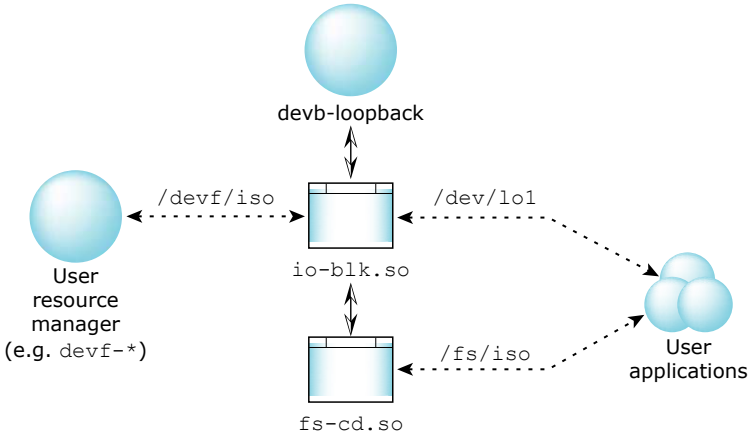
Internal function calls, normally targeted at a SCSI/CAM driver, are translated into standard *read()* and *write()* calls onto the host file descriptor, and are used to populate the data content of a pseudo block device, which appears to the system as a disk.

Within this framework, any resource-manager or file-descriptor object can be viewed as a block-special device and be mounted as a disk filesystem. For example, you can't normally directly use an ISO9660 image stored on a `devf-*` device, because `devf-*` doesn't load `fs-*.so` modules; by using `devb-loopback` to present this image file as a block device, you can then mount it using *fs-cd.so* (p. 787). For example:

```
devb-loopback fd=/devf/iso blk automount=lo0:/fs/iso:cd
```

This results in the following arrangement:

Copyright © 2014, QNX Software Systems Limited



Other uses might be to mount images from `/dev/shmem`, an image filesystem (IFS), or over NFS.

The host file needs to be pregrown. For example, assuming that `/fs/flash` is a mounted `devf` filesystem, you'd have to do the following setup first:

```
touch /fs/flash/q4.img
dinit -hq -S32m /fs/flash/q4.img
```

You could then use that file on flash as a read-write *fs-qnx4.so* (p. 820) filesystem:

```
devb-loopback blk cache=128k,auto=none loopback fd=/fs/flash/q4.img
mount -tqnx4 /dev/lo0 /q4flash
```

Using `devb-loopback` in this case adds functionality that a disk filesystem format offers (access times, hard links, etc.) to the `devf-*` filesystem. In other cases, `devb-loopback` lets you use the caching (e.g. names and sectors) that *io-blk.so* (p. 993) supports, but that the resource manager underlying the host file might not.

Driver support

The resource manager(s) implementing the pathnames specified by fd= operands must support the following standard interfaces:

open(), close()

The device must be able to open and close a file descriptor to it. The `O_RDONLY` and `O_RDWR` modes should be supported, as should the `SH_DENY` share flags and `O_SYNC` modifier (if those options are used from the `devb-loopback` command line); the default `libc` iofunc open handler is suitable.

fstat()

The device must support a `stat` method, which is used to determine the size and geometry of the underlying device. The default `libc` iofunc handler is suitable, provided there's an associated `iofunc_mount_t` which sets a suitable “blocksize” (the default is 1 byte, which isn't appropriate for use

Copyright © 2014, QNX Software Systems Limited

as a block device; it must be one of 512, 1024, 2048, or 4096). See also *devctl()* (below) and the loopback blksz= option (above).

read(), write()

The device must support read and write callouts, which are used to transfer "disk blocks" between the device and the filesystem buffer cache. I/O is in units of the advertised block size and within the bounds of the advertised device size. If the device supports the `_IO_XTYPE_OFFSET` modifier (corresponding to *pread()* and *pwrite()*), then that interface is used; otherwise a combined seek plus normal read/write is made.

lseek()

If `_IO_XTYPE_OFFSET` isn't supported (above), then the device must implement an `lseek` method. The default `libc` iofunc callout is suitable.

devctl()

The device may optionally implement the `DCMD_CAM_DEVINFO` *devctl()* command (refer to the `<sys/dcmd_cam.h>` and `<sys/cam_device.h>` header files). This allows the device to set certain low-level fields to fine-tune emulation behavior. The `DCMD_ALL_GETFLAGS` command is also used to probe the status of a (removable) file descriptor. However, suitable defaults for both these can be inferred from the *fstat()* query (above).

Mounting

You can mount the filesystem image via the loopback device (`/dev/lo*`) from the command line with the `mount` command:

```
devb-loopback blk cache=256k,vnode=128 \
              loopback ro,blksz=2048,fd=/mnt/EFS/tts.iso &
mount -r -tcd /dev/lo0 /mnt/tts
```

or via the blk automount= option:

```
devb-loopback blk cache=256k,vnode=128,automount=lo0:/mnt/tts:cd \
              loopback ro,blksz=2048,fd=/mnt/EFS/tts.iso &
```

You can use *umount* (p. 2009) to unmount the filesystem. The file descriptors to the underlying host image files aren't closed, and `/dev/lo*` remain present, until `devb-loopback` is terminated.

Copyright © 2014, QNX Software Systems Limited

devb-mvSata

Driver for Marvell 88SX50XX SATA interfaces (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-mvSata [cam option[,option]...]
            [mvSata option[,option]...]
            [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas (`,`) to separate the options. You can put the cam, mvSata, and blk groups of options in any order.

cam options

The cam options control the common access methods:

lun=*mask*

Enable Logical Unit Number (LUN) scanning for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is `0x00`.

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose: display full information about SCSI units (devices) on startup.

mvSata options

Copyright © 2014, QNX Software Systems Limited

The mvSata options control the driver's interface to the mvSata controller. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the mvSata keyword before each controller's set of options.

- Interface-specific options:

  irq=*req*

  The interrupt used by the controller.

  vid=*vid*

  The vendor ID of the controller.

  did=*did*

  The device ID of the controller.

  pci=*index*

  The PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

  nobmstr

  Don't use busmastering.

  port=*N*,*device*

  Specify options for *device* on port *N*.

  priority=*prio*

  Set the priority of the processing thread. The default is 21.

  timeout=*timeout*

  Set the I/O request timeout, in seconds. The default is 10.

  cache_line=*size*

  Set the PCI cache line size (16, 32, 64, or 128 bytes). The default is obtained from the PCI configuration.

- Device-specific options:

  geometry=*heads:cyl:sect*

  Specify the drive geometry.

  nobmstr

Copyright © 2014, QNX Software Systems Limited

Don't use busmastering.

multiblk=*blks*

Set multiblock mode, specifying the number of blocks per interrupt.

nonremovable

Report the device as being nonremovable.

chs

Use Cylinder-Head-Sector mode. The default is LBA.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword.

Description:

The `devb-mvSata` driver is for Marvell 88SX50XX SATA interfaces. It supports vendor ID `0x11ab` with at least the following device IDs:

Device ID	Chipset
5080	88SX5080
5081	88SX5081
5040	88SX5040
5041	88SX5041
6081	88SX6081
6041	88SX6041

Due to a chipset limitation, CD-ROM and DVD-ROM drives aren't supported.

Examples:

Detect all SATA controllers, and list all connected devices:

```
devb-mvSata &
```

Copyright © 2014, QNX Software Systems Limited

Files:

The `devb-mvSata` driver causes *io-blk.so* (p. 993) to adopt various block special devices under `/dev`. These devices are normally named `hd`*n*, where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
cam-disk.so (p. 91)	For hard-disk access.
`libcam.so`	Always

Exit status:

The `devb-mvSata` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The `devb-mvSata` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

Unless overridden with the blk automount= option (see *io-blk.so* (p. 993)), devices are mounted as:

Device	Mountpoint	Filesystem type
`/dev/hd0t77`	`/hd`	`qnx4`
`/dev/hd0t6`	`/dos`	`dos`
`/dev/hd0t11`	`/dos`	`dos`

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

Copyright © 2014, QNX Software Systems Limited

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-ram

Driver for RAM disk interface (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devb-ram [cam option[,option]...]
         [disk option[,option]...]
         [ram option[,option]...]
         [blk option[,option]...] &
```

Runs on:

QNX Neutrino

Options:

Use commas (`,`) to separate the options. You can put the cam, disk, ram, and blk groups of options in any order.

cam options

The cam options control the common access methods:

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose.

disk options

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword. For more information, see *cam-disk.so* (p. 91).

ram options

The ram options control the driver's interface to RAM:

address=*addr*

Copyright © 2014, QNX Software Systems Limited

The physical address to overlay. The default is not to overlay.

blksize=*size*

Set the sector size. The default is 512 bytes.

capacity=*blocks*

Specify the capacity of the RAM drive in the blocks of the size specified by the blksize option. The default is 4096 blocks (2 MB).

nodinit

Don't partition the RAM disk or format a QNX 4 filesystem on it.

blk options

The blk options control *io-blk.so* (p. 993). These options must follow the blk keyword and must be specified after any general or disk options. For more information, see *io-blk.so* (p. 993).

Description:

The `devb-ram` driver creates a RAM disk interface. When the capacity option isn't specified, `devb-ram` creates a 2 MB RAM disk.

By default, `devb-ram` partitions the RAM disk, leaving one block for the partition table itself, and making the remainder of the RAM disk (capacity minus 1) a t77 partition, which it then initializes (internally, not by spawning *dinit* (p. 626)) to have a blank *fs-qnx4.so* (p. 820) filesystem on it. If you specify the nodinit option, you can later manually format it, optionally partition the RAM disk with *fdisk* (p. 734) (but you can make the whole thing a filesystem), and then mount it.

By default, *io-blk.so* (p. 993) allocates 15% of system RAM for cache. The `devb-ram` system looks like a disk drive to `io-blk.so`, so it doesn't know that the cache is unnecessary. You should use the blk cache=512k option to reduce the cache size to the minimum.

Because `devb-ram` is a block device which reads from and writes to RAM, its operations go through a lot of layers before they actually get to RAM. For a RAM disk with better performance, use the blk ramdisk=... option to *io-blk.so* (p. 993). For more information, see “RAM disks” in the Connecting Hardware chapter of the QNX Neutrino *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

Examples:

Create a 4 MB RAM drive:

```
devb-ram ram capacity=8192 &
```

Files:

The `devb-ram` driver causes `io-blk.so` to adopt various block special devices under `/dev`. These devices are normally named `hd`*n*, where *n* is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
`cam-disk.so` (p. 91)	For RAM disk access.
`libcam.so`	Always

Exit status:

The `devb-ram` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The `devb-ram` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

Caveats:

While there's no limit to the size of a disk or partition, the limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. This I/O limit has no effect on the partition size for mounted filesystems. The maximum number of blocks is 2^{32}.

Known supported functions include:

chmod(), *chown()*, *close()*, *closedir()*, *creat()*, *devctl()*, *dup()*, *dup2()*, *fcntl()*, *fpathconf()*, *fstat()*, *lseek()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*, *opendir()*, *pathconf()*, *read()*, *readdir()*, *readlink()*, *rewinddir()*, *rmdir()*, *stat()*, *symlink()*, *unlink()* (not supported for directories), *utime()*, *write()*

Copyright © 2014, QNX Software Systems Limited

Note that certain calls (such as *pipe()*, as well as *read()* and *write()* on FIFOs) may require the *pipe* (p. 1555) manager.

Copyright © 2014, QNX Software Systems Limited

devb-umass

Driver for USB Mass Storage interface

In order to start this driver, you must be logged in as `root`, and the USB stack (*io-usb* (p. 1015)) needs to be running.

Syntax:

```
devb-umass [blk option[,option]...]
           [cam option[,option]...]
           [disk option[,option]...]
           [umass options[,option]...]
```

Runs on:

QNX Neutrino

Options:

Use commas (`,`) to separate the options. You can put the blk, cam, disk, and umass groups of options in any order.

blk options

The blk options control *io-blk.so* (p. 993). If specified, they must follow the blk keyword. For more information, see `io-blk.so`.

cam options

The cam options control the common access methods:

quiet

Be quiet: don't display any information on startup.

verbose

Be verbose: display full information about units (devices) on startup.

pnp

Enable CAM plug and play (i.e. don't exit at startup when no devices are found). The default is off.

disk *options*

Copyright © 2014, QNX Software Systems Limited

The disk options control the driver's interface to *cam-disk.so* (p. 91). If specified, they must follow the disk keyword.

umass options

The umass options control the driver's interface to the USB device. If you've installed multiple devices, you can repeat these options for each device. Remember, however, to specify the umass keyword *before* each controller's set of options.

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb/io-usb`.

wait=*num*

Wait *num* of seconds for the USB stack. The default is 60 seconds.

vid=*vid*

The vendor ID of the device.

did=*did*

The device ID of the device.

busno=*bus*

The bus number of the USB controller.

devno=*dev*

The USB address of the device.

iface=*if*

The particular interface number of the device.

priority=*prio*

Set the priority of the processing thread. The default is 21.

ignore_csw

Ignore the Command Status Wrapper. Some devices return invalid data for the CSW.

Description:

The `devb-umass` driver is the driver for a USB mass storage interface.

Copyright © 2014, QNX Software Systems Limited

Examples:

Assume a USB controller, and list all connected devices:

```
devb-umass &
```

Assume a USB controller, and list/wait for all connected devices:

```
devb-umass cam pnp &
```


Copyright © 2014, QNX Software Systems Limited

devc-con, devc-con-hid

Simple VGA console and keyboard I/O manager (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
devc-con [options] &
devc-con-hid [options] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-C *size*

Specify the size of the canonical buffer in bytes (default 256).

-E

Start in raw mode.

-e

Start in edited mode (the default).

-h

(`devc-con-hid` only) Don't connect to the *io-hid* (p. 1005) server; read from the keyboard controller instead.

-I *size*

Specify the size of the interrupt input buffer in bytes (default 2048).

-k

(`devc-con` and `devc-con-hid` only) Disable keyboard (don't attach a keyboard interrupt handler).

-L [P][N][C][S]

Copyright © 2014, QNX Software Systems Limited

Set the initial state of the keyboard and its LEDs (default all off):

- C — turn CapsLock on.
- P — preserve the keyboard state. This overrides all the other -L options.
- N — turn NumLock on.
- S — turn ScrollLock on.

-n *num_ports*

(`devc-con` and `devc-con-hid` only) The number of virtual console ports to create. The default is `4`; the maximum is `9`.

-O *size*

Specify the size of the interrupt output buffer in bytes (default 2048).

-o *opt*[,*opt*...]

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

-r *rate*[,*delay*]

Specify the initial keyboard typematic rate (in Hz) and, optionally, the initial keyboard delay (in ms). The default values are 30 Hz and 500 ms.

The keyboard typematic rate is the number of times per second that a depressed key repeats. On a PC/AT-compatible system, this ranges from 2 - 30 characters per second. If the option -r 0 is specified, the keyboard typematic rate isn't set by the driver.

The keyboard delay is defined as the time from when a key is first pressed to when the start of the first repeated key is generated. On a PC/AT-compatible system, the keyboard delay can range from 250 - 1000 milliseconds.

Description:

The `devc-con` manager provides an interface to the VGA console screen and keyboard.

For serial ports, use the appropriate *`devc-ser*`* (p. 169) manager.

Copyright © 2014, QNX Software Systems Limited

When devc-con starts, it creates and manages the devices /dev/con1, /dev/con2, and so on, up to the number of ports specified by the -n option.

The devc-con-hid manager is similar to devc-con, but works in conjunction with *io-hid* (p. 1005) and supports PS2, USB, and all other human-interface devices.

The devc-con-hid manager was added in QNX Momentics 6.3.0 Service Pack 3.

If you read from /dev/console, these managers return the characters typed on the keyboard; if you write to /dev/console, the managers write to the screen.

If your application uses /dev/console, you should create a link from it to one of /dev/con1, /dev/con2, ... by adding a line like this to the buildfile used by mkifs:

```
[type=link] /dev/console = /dev/con1
```

The devc-con and devc-con-hid managers all emulate an 80×25 ANSI terminal.

Keyboard control

You can use the keyboard to switch between virtual consoles.

Each virtual console can run different applications that use the entire screen. The keyboard is attached to the virtual console that's currently visible. You can switch from one virtual console to another — and thus from one application to another — by entering the following keychords:

If you want to see:	Press:
The next active console	**Ctrl**–**Alt**–**Enter** *or* **Ctrl**–**Alt**–**+** (plus)
The previous active console	**Ctrl**–**Alt**–**-** (minus)

The **+** (plus) and **-** (minus) keys used in the console-switching keychords are those found on the numeric keypad.

You can also jump to a specific console by using the **Ctrl**–**Alt**–**n**, where *n* is a numeric digit that represents the console number of a virtual console. For instance:

If you want to see:	Press:
/dev/con1	**Ctrl**–**Alt**–**1**
/dev/con2 (if available)	**Ctrl**–**Alt**–**2**
...	...

Copyright © 2014, QNX Software Systems Limited

If you want to see:	Press:
`/dev/con10` (if available)	**Ctrl**–**Alt**–**0**

Character sets

The `devc-con` and `devc-con-hid` managers let you choose the character sets in use from a "palette" of character sets, each of which is independently programmable to contain one of several builtin character sets.

The in-use range of characters is divided into four regions which span character numbers (in hexadecimal) `0x00` through `0xff`. Two of these regions are fixed sets of control characters, while the other two are configurable to contain a choice of character sets:

Hex:	Name:	May be set to one of:
0x00-0x1f	`C0` (Control Zero)	Not configurable
0x20-0x7f	`GL` (Graphics Left)	`G0`, `G1`, `G2`, `G3`
0x80-0x9f	`C1` (Control One)	Not configurable
0xa0-0xff	`GR` (Graphics Right)	`G1`, `G2`, `G3`

You can set each of the `GL` and `GR` in-use character sets to a choice of several character sets from the `G0`, `G1`, `G2` and `G3` character sets.

The screen control codes to set `GL` and `GR` are as follows:

To set:	To:	Use:
`GL`	`G0`	{LS0} = {SI} (0f)
`GL`	`G1`	{LS1} = {SO} (0e)
`GL`	`G2`	{LS2} = {ESC n} (1b 6e) or {SS2} (8e)
`GL`	`G3`	{LS3} = {ESC o} (1b 6f) or {SS3} (8f)
`GR`	`G1`	{LS1R}= {ESC ~} (1b 7e)
`GR`	`G2`	{LS2R}= {ESC }} (1b 7d)
`GR`	`G3`	{LS3R}= {ESC |} (1b 7c)

The {LS*} codes stand for "Locking Shift". When character sets are selected by these means, they remain in effect until another {LS*} code is sent.

Copyright © 2014, QNX Software Systems Limited

The {SS*} codes stand for "Single Shift" and affect the next character only. After that character, the character set in effect reverts to its previous setting. There are only two {SS*} codes, {SS2} and {SS3} which maps G2 into GL and G3 into GL, respectively.

The G0 through G3 characters sets may each be set to any of the available builtin fonts. The control code to do this is:

```
ESC g s
```

Where:

g:	Sets:
(	G0
)	G1
*	G2
+	G3

And:

s:	Specifies:
B	ASCII
0	Special (DEC Graphic)
<	ISO-Latin1 Supplemental
U	PC Character Set

Character set defaults

The in-use character sets are as follows:

In-use char set:	Defaults to:
GR	G2
GL	G0

The character set codes are as follows:

Char set:	Defaults to:
G0	ASCII charset
G1	Special charset (DEC Graphic)
G2	Supplemental charset (ISO-Latin 1)
G3	Special charset (DEC Graphic)

Copyright © 2014, QNX Software Systems Limited

Character set example:

Set the `GL` in-service character set (0x20-0x7f) to the PC character set through `G1`, write some characters, then switch `GL` back to `G0`:

```
{ESC )U}  1e 29 55    (Set G1 to be the PC character set)
{SO}      0e          (Set GL to G1)
.
.                     (Write chars in PC graphics char set)
.
{SI}      0f          (Set GL to G0)
```

DECIMAL VALUE ➡		0	16	32	48	64	80	96	112
⬇	HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

Figure 5: The PC character set 0x00-0x7f.

Copyright © 2014, QNX Software Systems Limited

Figure 6: The PC character set 0x80-0xff.

ANSI screen control codes

Note the following abbreviations used in the tables:

(220+)

A VT220 Level 2 function

(NA)

Not ANSI standard

(NI)

Not implemented

(NFI)

Not fully implemented

C0 control codes

The C0 control codes are as follows:

ASCII	ANSI Mnemonic	Hex	Action
{NUL}		00	Null

Copyright © 2014, QNX Software Systems Limited

ASCII	ANSI Mnemonic	Hex	Action
{BEL}		07	Bell
{BS}		08	Back Space (VT100 defaults to no wrap from left margin)
{HT}		09	Horizontal Tab (VT100 defaults to no autowrap)
{LF}		0A	Linefeed or Newline
{VT}		0B	Same as LF
{FF}		0C	Clears Screen (QNX Extension)
{CR}		0D	Move cursor to left margin
{SO}	{LS1}	0E	`GL` is set to `G1`
{SI}	{LSO}	0F	`GL` is set to `G0` (default)
{XON}	{DC1}	11	XON
{XOFF}	{DC0}	13	XOFF
{CAN}		18	Cancels ESC sequence
{SUB}		1A	Cancels ESC sequence and prints ?
{ESC}		1B	Start of ESC sequence
{DEL}		7F	Ignored on output

ESC control sequences

The ESC control sequence codes are as follows:

String	Hex	Action
{ESC 7}	1B 37	Save cursor
{ESC 8}	1B 38	Restore cursor

Copyright © 2014, QNX Software Systems Limited

String	Hex	Action
{ESC =}	1B 3D	Set application keypad mode
{ESC >}	1B 3E	Set numeric keypad mode (default)
{ESC D}	1B 44	7-bit codes for {IND} (84)
{ESC E}	1B 45	7-bit codes for {NEL} (85)
{ESC H}	1B 48	7-bit codes for {HTS} (88)
{ESC M}	1B 4D	7-bit codes for {RI} (8D)
{ESC N}	1B 4E	7-bit codes for {SS2} (8E)
{ESC O}	1B 4F	7-bit codes for {SS3} (8F)
{ESC P}	1B 50	7-bit codes for {DCS} (90)
{ESC [}	1B 5B	7-bit codes for {CSI} (9B)
{ESC \}	1B 5C	7-bit codes for {ST} (9C)
{ESC]}	1B 5D	7-bit codes for {OSC} (9D)
{ESC ^}	1B 5E	7-bit codes for {PM} (9E)
{ESC _}	1B 5F	7-bit codes for {APC} (9F)
{ESC Z}	1B 5A	Identify terminal
{ESC c}	1B 63	Hard Reset (clears screen) (use {CSI ! P} for soft reset)
{ESC n}	1B 6E	(LS2) `GL` is set to `G2` (220+)
{ESC o}	1B 6F	(LS3) `GL` is set to `G3` (220+)
{ESC |}	1B 7C	(LS3R) `GR` is set to `G3` (220+)
{ESC }}	1B 7D	(LS2R) `GR` is set to `G2` (220+) (default)
{ESC ~}	1B 7E	(LS1R) `GR` is set to `G1`
{ESC sp F}	1B 20 46	Keyboard generates 7-bit `C1` codes (incl. CSI) (default)

Copyright © 2014, QNX Software Systems Limited

String	Hex	Action
{ESC sp G}	1B 20 47	Keyboard generates 8-bit C1 codes (incl. CSI) (220+)
{ESC (0}	1B 28 30	Set G0 to special charset
{ESC (<}	1B 28 3C	Set G0 to supplemental charset
{ESC (A}	1B 28 41	Set G0 to U.K. charset (Not implemented; same as ASCII)
{ESC (B}	1B 28 42	Set G0 to ASCII charset (default)
{ESC (U}	1B 28 55	Set G0 to PCterm graphics
{ESC) 0}	1B 29 30	Set G1 to special charset (default)
{ESC) <}	1B 29 3C	Set G1 to supplemental charset
{ESC) A}	1B 29 41	Set G1 to U.K. charset (NI; same as ASCII)
{ESC) B}	1B 29 42	Set G1 to ASCII charset
{ESC) U}	1B 29 55	Set G1 to PCterm graphics
{ESC * 0}	1B 2A 30	Set G2 to special charset (220+)
{ESC * <}	1B 2A 3C	Set G2 to supplemental charset (220+) (default)
{ESC * B}	1B 2A 42	Set G2 to ASCII charset (220+)
{ESC * U}	1B 2A 55	Set G2 to PCterm graphics
{ESC + 0}	1B 2B 30	Set G3 to special charset (220+) (default)
{ESC + <}	1B 2B 3C	Set G3 to supplemental charset (220+)
{ESC + B}	1B 2B 42	Set G3 to ASCII charset (220+)
{ESC + U}	1B 2B 55	Set G3 to PCterm graphics

Copyright © 2014, QNX Software Systems Limited

C1 control characters (220+)

You can do any 8-bit C1 code with 7-bit ESC followed by the 8-bit code minus 0x40 hex. For instance, you can represent the CSI (control sequence introducer) in 8-bit mode as 0x9b, while in 7-bit mode you must express it as ESC [(0x1b 0x5b).

The C1 control character codes are as follows:

ASCII	Hex	Action
{IND}	84	Move cursor down with scroll
{NEL}	85	Move to left margin on next line with scroll
{HTS}	88	Set horizontal tab
{RI}	8D	Move cursor up with scroll
{SS2}	8E	GL is set to G2 for 1 character
{SS3}	8F	GL is set to G3 for 1 character
{DCS}	90	Start of Device control string
{CSI}	9B	Control sequence introducer
{ST}	9C	End of Device control string
{OSC}	9D	Operating System Command
{PM}	9E	Privacy Message
{APC}	9F	Application Program Command

CSI control sequences

In 7-bit mode, CSI is ESC [. In 8-bit mode, CSI is (hex)0x9B. Use the ANSI specification to represent the variable *n*, e.g. to print two spaces:

```
printf( "%c%c", 0x9b, 0x32 ) ;
```

The CSI control sequence codes are as follows:

Copyright © 2014, QNX Software Systems Limited

ASCII	Hex	Action
{CSI [*n*] @}	9B [*n*] 40	Insert *n* spaces at cursor (default = 1 space)
{CSI [*n*] A}	9B [*n*] 41	Cursor up *n* rows, no wrap (default = 1 row)
{CSI [*n*] B}	9B [*n*] 42	Cursor down *n* rows, no wrap (default = 1 row)
{CSI [*n*] C}	9B [*n*] 43	Cursor right *n* columns, no wrap (default = 1 column)
{CSI [*n*] D}	9B [*n*] 44	Cursor left *n* columns, no wrap (default = 1 column)
{CSI [*n*] F}	9B [*n*] 46	Cursor up *n* rows, positioned in first column (default = 1 row)
{CSI [*n*] G}	9B [*n*] 47	Move cursor to column *n* (default = column 1)
{CSI [*r*[;*c*]] H}	9B [*r* [3B *c*]] 48	Cursor position (default = row 1; column 1)
{CSI [*n*] J}	9B [*n*] 4A	Erase 0=cur-EOS 1=HOME-cur 2=screen (default = 0 (to end of screen))
{CSI [*n*] K}	9B [*n*] 4B	Erase 0=cur-EOL 1=BOL-cur 2=line (default = 0 (to end of line))
{CSI [*n*] L}	9B [*n*] 4C	Insert *n* lines (default = 1 line)
{CSI [*n*] M}	9B [*n*] 4D	Delete *n* lines (default = 1 line)
{CSI [*n*] P}	9B [*n*] 50	Delete *n* chars (default = 1 char)
{CSI [*n*] S}	9B [*n*] 53	Scroll forward *n* lines (default = 1 line)
{CSI [*n*] T}	9B [*n*] 54	Scroll backward *n* lines (default = 1 line)

Copyright © 2014, QNX Software Systems Limited

ASCII	Hex	Action
{CSI [*n*] X}	9B [*n*] 58	Erase cur for *n*-1 chars (default = 1 (0 chars))
{CSI Z}	(9B 5A)	Back tab
{c CSI [*n*] b}	c 9B [*n*] 62	Repeat **GR** or **GL** character *c*, *n* times. *c* is the last displayable character; *n* defaults to 1 time.
{CSI 0 c}	(9B 30 63)	Primary device attrib request
{CSI [*n*] d}	9B [*n*] 64	Move cursor to line n (default = line 1)
{CSI [*n*] g}	9B [*n*] 67	Tab clear 0=cursor 2=all (default = 0)
{CSI [*n*[;*n*]...] h}	9B [*n*[3B *n*]...] 68	Standard Set mode (See modes table) (default=none)
{CSI ? [*n*[;*n*]...] h}	9B 3F [*n*[3B *n*]...] 68	Private Set mode (See modes table) (default=none)
{CSI [*n*[;*n*]...] l}	9B [*n*[3B *n*]...] 6C	Standard Reset mode (See modes table) (default=none)
{CSI ? [*n*[;*n*]...] l}	9B 3F [*n*[3B *n*]...] 6C	Private Reset mode (See modes table) (default=none)
{CSI [*n*[;*n*]...] m}	9B [*n*[3B *n*]...] 6D	Select Graphic Rendition (See below) (default = 0)
{CSI *n* n}	9B *n* 6E	Device status 5=status 6=cursor/pos
{CSI [*r*[;*c*]] r}	9B [*r* [3B *c*]] 72	Set scroll region and home cursor
{CSI r}	9B 72	Disable scroll region & home cursor
{CSI s}	9B 73	Save cursor

Copyright © 2014, QNX Software Systems Limited

ASCII	Hex	Action
{CSI u}	9B 75	Restore cursor
{CSI ! p}	9B 21 70	Soft reset
{CSI [*n*[;*n*]]]}	9B [*n* [3B *n*]...] 5D	Set default 1=underline 2=half-intensity 8=colorset (default=none)
{CSI = [*f* [;*d*]] B}	9B 3D [*f* [3B *d*]] 46	Set frequency(Hz) and duration (ms) for bell (default=100Hz, 250ms)
{CSI = [*n*] F}	9B 3D [*n*] 46	Set and Save foreground color
{CSI = [*n*] G}	9B 3D [*n*] 47	Set and Save background color

Graphic rendition

The graphic rendition codes are as follows:

Number	Meaning
0	All attributes off (except charset (10, 11, 12))
1	Bold
2	Half intensity (default to cyan on color screen)
4	Underline (default to red on color screen)
5	Blink
7	Reverse
9	Invisible
10	Exit alternate char set (`GR` & `GL` are restored)
11	Enter PC-lower char set (`GR` & `GL` are ASCII; `C0` & `C1` are PC_LO except for ESC)
12	Enter PC-higher char set (`GR`, `C1` & `GL`, `C0` are PC_HI except for ESC)
21	Normal intensity (un-Bold)

Copyright © 2014, QNX Software Systems Limited

Number	Meaning
22	Normal intensity (un-Half intensity)
24	Disable underline
25	Disable blink
27	Disable reverse
29	Visible
30-37	Set foreground color (30+*color_number*, see below)
39	Set foreground to saved
40-47	Set background color (40+*color_number*, see below)
49	Set background to saved

Color numbers

The color codes are as follows:

color_number	Description
0	Black
1	Red
2	Green
3	Brown
4	Blue
5	Violet
6	Cyan
7	White

Modes

Modes are as follows:

Mode string	Description
?1h	cursor key = Application
?1l	cursor key = ANSI (default)
?3h	132 column (Not Implemented)

Copyright © 2014, QNX Software Systems Limited

Mode string	Description
?3l	80 column (default)
?5h	Reverse screen
?5l	Not Reverse screen (default)
?6h	Origin mode
?6l	Absolute mode
?7h	Auto wrap on
?7l	Auto wrap off (default)
?25h	Visible cursor (default)
?25l	Invisible cursor
?45h	Reverse wrap-around mode
?45l	No reverse wrap-around
?66h	keypad = Application
?66l	keypad = ANSI
?67h	Backspace key generates BS
?67l	Backspace key generates DEL

Mapping from QNX keyboard to ANSI keys

The ANSI key mapping codes are as follows:

Key	Normal	Shift	Ctrl	Alt
Enter	CR	CR	CR	CR
Tab	TAB	CSI Z	CSI z	
BS	BS	RUB	RUB	BS
ESC	ESC	ESC	ESC	ESC
F1	SS3 P	SS3 p	CSI 1~	CSI 17~
F2	SS3 Q	SS3 q	CSI 2~	CSI 18~
F3	SS3 R	SS3 r	CSI 3~	CSI 19~
F4	SS3 S	SS3 s	CSI 4~	CSI 20~
F5	SS3 T	SS3 t	CSI 5~	CSI 21~
F6	SS3 U	SS3 u	CSI 6~	CSI 22~

Copyright © 2014, QNX Software Systems Limited

Key	Normal	Shift	Ctrl	Alt
F7	SS3 V	SS3 v	CSI 7~	CSI 23~
F8	SS3 W	SS3 w	CSI 8~	CSI 24~
F9	SS3 X	SS3 x	CSI 9~	CSI 25~
F10	SS3 Y	SS3 y	CSI 10~	CSI 26~
F11	SS3 Z	SS3 z	CSI 11~	CSI 27~
F12	SS3 A	SS3 a	CSI 12~	CSI 28~
Home	CSI H		CSI h	CSI H
	CSI A		CSI a	CSI A
PgUp	CSI V		CSI v	CSI V
Minus	CSI S		CSI s	CSI S
	CSI D		CSI d	CSI D
kpd 5	CSI G		CSI g	CSI G
	CSI C		CSI c	CSI C
Plus	CSI T		CSI t	CSI T
End	CSI Y		CSI y	CSI Y
	CSI B		CSI b	CSI B
PgDn	CSI U		CSI u	CSI U
Ins	CSI @		CSI `	CSI @
Del	CSI P		CSI p	CSI P
Prt	NOP	NOP	NOP	NOP
SysRq	NOP	NOP	NOP	NOP
a	a	A	SOH	SS2 a
b	b	B	STX	SS2 b
c	c	C	ETX	SS2 c
d	d	D	EOT	SS2 d
e	e	E	ENQ	SS2 e
f	f	F	ACK	SS2 f
g	g	G	BEL	SS2 g

Copyright © 2014, QNX Software Systems Limited

Key	Normal	Shift	Ctrl	Alt
h	h	H	BS	SS2 h
i	i	I	HT	SS2 i
j	j	J	LF	SS2 j
k	k	K	VT	SS2 k
l	l	L	FF	SS2 l
m	m	M	CR	SS2 m
n	n	N	SO	SS2 n
o	o	O	SI	SS2 o
p	p	P	DLE	SS2 p
q	q	Q	DC1	SS2 q
r	r	R	DC2	SS2 r
s	s	S	DC3	SS2 s
t	t	T	DC4	SS2 t
u	u	U	NAK	SS2 u
v	v	V	SYN	SS2 v
w	w	W	ETB	SS2 w
x	x	X	CAN	SS2 x
y	y	Y	EM	SS2 y
z	z	Z	SUB	SS2 z

International keyboard layouts

The `devc-con` and `devc-con-hid` managers support international keyboard layouts. By default, they use the original US-101 layout.

If the file `/etc/kbd.tbl` is present when you start `devc-con` or `devc-con-hid`, it's loaded and used instead. You can reload this file at runtime by pressing **Ctrl**–**Alt**–**Space**. (If you're using VMWare, you may need to press this twice).

This, like all other key combinations, is subject to configuration! However, it isn't possible to redefine compose sequences.

We provide the following layout files, in ***${QNX_TARGET}***`/etc/`:

`kbd.tbl.de`

Copyright © 2014, QNX Software Systems Limited

DE-102 (German) layout.

kbd.tbl.us

US-101 layout (the default).

The keyboard-layout file's structure is very simple and rigid. It must contain either *exactly* 5 × 96 or *exactly* 6 × 96 hexadecimal entries, separated by whitespace, newlines, or commas:

- If the file contains 5 × 96 entries, the left and right **Alt** keys are both treated as regular **Alt** keys.
- If the file has 6 × 96 entries, the right **Alt** key is treated as an **AltGr** key, and the last 96 entries must define key codes for each key with **AltGr** pressed.

Entries must be no longer than 4 hex digits (16 bits) each. Comments start with a number sign (#) and extend to the end of the line.

Each run of 96 entries defines the semantics of up to 96 different keys under certain conditions:

Entries:	Define semantics for keys:
000–095	Without any modifiers
096–191	With **Shift** pressed
192–287	With **Ctrl** pressed
288–383	With **Alt** pressed
384–479	With **Ctrl**–**Alt** pressed
480–575	With **AltGr** (right **Alt**) pressed

The 96 entries in a run are indexed by keyboard scan codes. You'll need to know those scan codes in order to make up your own keyboard definition. Given below is the scancode/symbol-mapping for a US-101 keyboard:

```
 0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
   ,  Esc,  '1',  '2',  '3',  '4',  '5',  '6' '7',  '8',  '9',  '0',  '-',  '=',  Rub,  Tab  ; 00
'q',  'w',  'e',  'r',  't',  'y',  'u',  'i' 'o',  'p',  '[',  ']',  Ent,  Ctl,  'a',  's'  ; 10
'd',  'f',  'g',  'h',  'j',  'k',  'l',  ';' ''',  '',   Shf,  '\',  'z',  'x',  'c',  'v'  ; 20
'b',  'n',  'm',  ',',  '.',  '/',  Rsh,  '*' Alt,   SP,  Cap,   F1,   F2,   F3,   F4,   F5  ; 30
 F6,   F7,   F8,   F9,  F10,  Num,  Scr,  Hom  Up,  PgU,   K-,  Lft,   K5,  Rig,   K+,  End  ; 40
Dwn,  PgD,  Ins,  Del,    ,    ,    ,  F11  F12,    ,    ,    ,    ,    ,    ,       ; 50
```

For every scan code, two bytes of data are given by the entries. The high byte defines a number of flags for the key (see below), while the low byte usually carries the actual data to be given to the user when the key is pressed. For Shift, Lock, and special keys, the low byte carries additional, function-dependent information (see below).

The entries' high bits are as follows:

- 00 — Data key
- 01 — Function key

Copyright © 2014, QNX Software Systems Limited

- 02 — Shift key:
  - 0201 — **Shift**
  - 0202 — **Ctrl**
  - 0204 — **Alt**
  - 0208 — **Rshift**
- 04 — **NumLock**-dependent
- 08 — Lock key:
  - 0801 — **ScrollLock**
  - 0802 — **NumLock**
  - 0804 — **CapsLock**
- 10 — Dead key
- 20 — **CapsLock**-dependent
- 40 — Special:
  - 4001 — reboot (**Ctrl**–**Alt**–**Del**)
  - 4002 — debug (**Ctrl**–**Alt**–**Esc**)
  - 4003 — next console (**Ctrl**–**Alt**–**+** or **Ctrl**–**Alt**–**Enter**)
  - 4004 — previous console (**Ctrl**–**Alt**–**-**)
  - 4005 — console 1 (**Ctrl**–**Alt**–**1**)
  - 4006 — console 2 (**Ctrl**–**Alt**–**2**)

    ...
  - 400C — console 8 (**Ctrl**–**Alt**–**8**)
  - 400D — console 9 (**Ctrl**–**Alt**–**9**)
  - 400E — console 10 (**Ctrl**–**Alt**–**0**)
  - 400F — increase font size (**Ctrl**–**Alt**–**>**)
  - 4010 — decrease font size (**Ctrl**–**Alt**–**<**)
  - 4011 — help (**Ctrl**–**Alt**–**?**)
  - 4012 — break (**Ctrl**–**Break**)
  - 4013 — hang up (**Ctrl**–**Alt**–**End**)
  - 4020 — print screen (**Ctrl**–**Alt**–**PrtScn**)
  - 4021 — hot1 (**Ctrl**–**Alt**–**F1**)
  - 4022 — hot2 (**Ctrl**–**Alt**–**F2**)

    ...
  - 402B — reload (**Ctrl**–**Alt**–**Space**)
- 80 — invalid key

Copyright © 2014, QNX Software Systems Limited

Examples:

Typical command line:

```
devc-con -n4
```

Files:

/dev/con1, /dev/con2, ...

Console port devices.

/etc/kbd.tbl

The keyboard layout; see "*International keyboard layouts* (p. 282)," above.

Errors:

If an error occurs, the keyboard doesn't work in text mode.

Copyright © 2014, QNX Software Systems Limited

devc-par

Parallel port manager (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devc-par [options] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-b *port*

Which BIOS port to use (1-4). Don't use this option with the -p option.

-p *address*

Base I/O address of the parallel port. The I/O port address may be specified in hexadecimal form (e.g. `0x140`) or octal form (e.g. 0140) as well as in decimal. Don't use this option with the -b option.

-N *name*

Register the parallel devices using *name* as the name (defaults to `par`, giving `/dev/par`).

-O *size*

Size of the output buffer in bytes (default 1000).

-o *opt*[,*opt*...]

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.

Copyright © 2014, QNX Software Systems Limited

- priority=*prio* — set the working priority of the internal pulse.

-P *priority*

Priority of the writer agent task. Because the writer agent polls the parallel ports, it should run at a lower priority than most other tasks. The default priority is 9.

-s *number*

Number of times to check a busy printer before resorting to a 100 ms sleep. (Default is -1, always sleep if the printer is busy.)

Description:

The `devc-par` manager is a parallel port manager for the QNX Neutrino RTOS. It can support up to 4 parallel ports.

The `devc-par` driver polls the hardware to detect if a character has been sent.

If you don't specify any ports (using the -p or -b options), `devc-par` tries to interrogate the BIOS area to determine the number of parallel ports detected by the BIOS. If no ports are found, `devc-par` silently exits.

You can use the -p option to override the use of the BIOS data area (at `0040:0008`).

The only translation of output is the mapping of a newline character to a CR/LF if the OPOST flag is set.

Reading from `devc-par` works the same as reading from `/dev/null`.

Examples:

Start `devc-par`, defaulting for all parallel ports found by the BIOS:

```
devc-par &
```

Start `devc-par` with only the first parallel port:

```
devc-par -p 0x3f8 &
```

Or:

```
devc-par -b 1 &
```

Copyright © 2014, QNX Software Systems Limited

devc-pty

Pseudo-tty communications manager (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
devc-pty [options] &
```

Runs on:

QNX Neutrino

Options:

-C *size*

Specify the size of the canonical buffer in bytes (default 256).

-I *size*

Size of input buffer in bytes (default 256).

-n *numptys*

Create *numptys* ptys (default 8).

-O *size*

Size of output buffer in bytes (default 3 × 512).

-o *opt*[,*opt*...]

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

Description:

The `devc-pty` manager is a small pseudo-tty manager for the QNX Neutrino RTOS. It can support up to 256 ptys, using the naming scheme:

Copyright © 2014, QNX Software Systems Limited

- `pty[p-zP-T][0-9a-f]` for the master device
- `tty[p-zP-T][0-9a-f]` for the slave device

The master and slave device pair share the same letter and hexadecimal digit.

Examples:

Start `devc-pty` with 32 ptys:

```
devc-pty -n 32 &
```

Copyright © 2014, QNX Software Systems Limited

devc-ser8250

8250 serial communications manager (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devc-ser8250 [[options]
              [port[^shift][,intr]]]... &
```

Runs on:

QNX Neutrino

Targets:

ARMv7 and x86 hardware with an 8250-compatible UART

Options:

The options are position-dependent and affect the subsequent ports.

-b ***number***

The initial baud rate (default 57600).

-C ***size***

The size of the canonical buffer in bytes (default 256).

-c ***clock*****[/***divisor***]**

Define a custom clock rate, in hertz, and divisor for the serial port. The default (-c 1843200/16) is suitable for compatible PC serial ports.

-E

Start in raw mode (the default). Software flow control is disabled by default.

-e

Start in edited mode (default raw). Software flow control is enabled by default.

-F

Copyright © 2014, QNX Software Systems Limited

Disable hardware flow control (default to hardware flow control enabled). Hardware flow control is not supported in edited mode.

-f

Enable hardware flow control (default). Hardware flow control is not supported in edited mode.

-I ***number***

The size of the interrupt input buffer in bytes (default 2048).

-O ***number***

The size of the interrupt output buffer in bytes (default 2048).

-o ***opt*****[,*****opt*****...]**

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

-Sls

Disable / enable software flow control. The default depends on the mode: in raw mode (-E, the default), it's disabled; in edited mode (-e), it's enabled.

The order in which you specify the -E or -e, and -S or -s options matters:

Options	Mode	Software flow control
-e	Edited	Enabled
-S -e	Edited	Enabled
-e -S	Edited	Disabled
-E	Raw	Disabled
-s -E	Raw	Disabled
-E -s	Raw	Enabled

-T ***number***

Enable the transmit FIFO and set the number of characters to be transmitted at each TX interrupt to 1, 4, 8, or 14. The default is 0 (FIFO disabled).

-t ***number***

Copyright © 2014, QNX Software Systems Limited

Enable the receive FIFO and set its threshold to 1, 4, 8, or 14 characters. The default is 0 (trigger disabled).

-u *number*

Append *number* to the device name prefix (`/dev/ser`). The default is `1`; additional devices are given increasing numbers.

-v[v]...

Be verbose; additional `v` characters cause more verbosity.

port

The hex I/O address (for x86 systems) of a serial port.

shift

The spacing of the device registers as a power of 2. For example:

`0`

Registers are 1 byte apart.

`1`

Registers are 2 bytes apart.

`2`

Registers are 4 bytes apart.

…

n

Registers are 2^n bytes apart.

The default *shift* is `0`.

intr

The interrupt used by this port; specified in hex if prefixed with `0x`, otherwise it's decimal.

Description:

The `devc-ser8250` manager is a small serial device manager for the QNX Neutrino RTOS. It can support any number of serial ports using 8250s, 14450s or 16550s. Each device can be assigned its own interrupt, or share an interrupt if the hardware supports interrupt sharing. If you don't specify any I/O ports for devices on an x86

Copyright © 2014, QNX Software Systems Limited

system, `devc-ser8250` assumes you want to use the standard PC ports of COM1 (3f8,4) and COM2 (2f8,3).

The serial driver's priority floats to the priority of the client. All internal events are processed at priority 24 (inherited from the internal pulse). The event handling priority is hard coded and isn't configurable by any of the options listed. (The driver's `main.c` program would need modification in order to change the priority).

When the driver talks to a client application, it's running at the priority of the client. All other processing takes place either at priority 24r or at interrupt time.

Each device is given a name in the pathname space of `/dev/ser`*n*, where *n* starts at 1 (unless changed via the -u option) and increases. If you use the default PC serial ports, this results in:

Device	Port	Interrupt
`/dev/ser1`	3f8	4
`/dev/ser2`	2f8	3

If your application uses `/dev/console`, you should create a link from it to one of `/dev/ser1`, `/dev/ser2`, ... by adding a line like this to the buildfile used by `mkifs`:

```
[type=link] /dev/console = /dev/ser1
```

All devices are fully interrupt driven and by default support standard hardware flow control on input and output (RTS/CTS). This can be disabled by the -F option.

Hardware flow control is not supported in edited mode.

A read request by default returns when at least 1 character is available. To increase efficiency, you can control three parameters to control when a read is satisfied:

Time

Return after a specified amount of time has elapsed.

Min

Return when this number of characters are in the input buffer.

Char

Return if this forwarding character is in the input buffer.

Copyright © 2014, QNX Software Systems Limited

If the Min value is greater than the size of the input buffer, the Min value is clipped to the size of the buffer. To avoid this, the size of the input buffer can be changed with the -I option.

These parameters are set using library routines (see *tcgetattr()*, *tcsetattr()*, *readcond()* and *TimerTimeout()* in the *C Library Reference*).

The `devc-ser8250` manager supports both raw and edited modes, making it a real tty device.

The following fields and flags are supported in the `termios` structure:

Field	Supported flags
c_cc	All characters
c_iflag	BRKINT ICRNL IGNBRK IXON
c_oflag	OPOST
c_cflag	CLOCAL CSIZE CSTOPB PARENB PARODD
c_lflag	ECHO ECHOE ECHOK ECHONL ICANON IEXTEN ISIG NOFLSH

Examples:

Start `devc-ser8250`, defaulting for COM1 and COM2:

```
devc-ser8250 &
```

Start `devc-ser8250`, defaulting for COM1 and COM2, but change baud rate to 38400 from 57600 default:

```
devc-ser8250 -b 38400 &
```

Start `devc-ser8250` with five ports (the last four preset to 38400 baud):

```
devc-ser8250 3f8,4 -b 38400 280,3 288,3 290,3 298,3 &
```

You can specify multiple -F and -f options; they're position-dependent, and affect the next serial port:

```
devc-ser8250 -F 3f8,4 -f 2f8,3 &
```


Copyright © 2014, QNX Software Systems Limited

devc-serpci

Driver for serial PCIs

You must be `root` to start this driver.

Syntax:

```
devc-serpci [vid=vid,did=did[,pci=pci]] [options] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-b ***number***

The initial baud rate (default 57600).

-C ***size***

The size of the canonical buffer in bytes (default 256).

-c ***clock*****[/***divisor***]**

Define a custom clock rate, in hertz, and divisor for the serial port. The default (-c 1843200/16) is suitable for compatible PC serial ports.

did=*did*

The PCI device ID, in hexadecimal (0x*XXXX*).

-E

Set options to “raw” (default).

-e

Start in edited mode (default raw). Software flow control is enabled by default.

-F

Copyright © 2014, QNX Software Systems Limited

Disable hardware flow control (default to hardware flow control enabled). Hardware flow control is not supported in edited mode.

-f

Enable hardware flow control (default). Hardware flow control is not supported in edited mode.

-I ***number***

The size of the raw input buffer in bytes (default 2048).

-O ***number***

("Oh") The size of the interrupt output buffer in bytes (default 2048).

-o ***opt*****[,*****opt*****...]**

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

pci=*pci*

The PCI index, in decimal (*XX*).

-Sls

Disable / enable software flow control. The default depends on the mode: in raw mode (-E, the default), it's disabled; in edited mode (-e), it's enabled.

The order in which you specify the -E or -e, and -S or -s options matters:

Options	Mode	Software flow control
-e	Edited	Enabled
-S -e	Edited	Enabled
-e -S	Edited	Disabled
-E	Raw	Disabled
-s -E	Raw	Disabled
-E -s	Raw	Enabled

-t ***number***

Copyright © 2014, QNX Software Systems Limited

Enable the receive FIFO and set its threshold to 1, 4, 8, or 14 characters. The default is 0 (trigger disabled).

-u *number*

Append *number* to the device name prefix (`/dev/ser`). The default is `1`; additional devices are given increasing numbers.

-v[v]...

Be verbose; additional `v` characters cause more verbosity.

vid=*vid*

The PCI vendor ID, in hexadecimal (0x*XXXX*).

Description:

The `devc-serpci` manager is a small serial device manager for the QNX Neutrino RTOS. This driver supports 16C*xxx* compatible Serial PCI cards.

The boards must use PCI I/O space for the registers, and the ports must be contiguous in memory.

Copyright © 2014, QNX Software Systems Limited

devc-serusb

Driver for USB-to-serial adaptors and USB CDC ACM devices

You must be `root` to start this driver.

Syntax:

```
devc-serusb [options]
```

Runs on:

QNX Neutrino

Targets:

ARMLE-v7, x86

Options:

-b ***number***

The initial baud rate (default 57600).

-C ***size***

The size of the canonical buffer, in bytes (default 256).

-d ***args*****[,*****args*** **...]**

Device-specific options. See below.

-E

Set options to “raw” (default).

-e

Start in edited mode (default raw). Software flow control is enabled by default.

-F

Disable hardware flow control (default to hardware flow control enabled). Hardware flow control is not supported in edited mode.

-f

Copyright © 2014, QNX Software Systems Limited

Enable hardware flow control (default). Hardware flow control is not supported in edited mode.

-I ***number***

The size of the raw input buffer in bytes (default 2048).

-O ***number***

("Oh") The size of the interrupt output buffer in bytes (default 2048).

-o ***opt*****[,*****opt*****...]**

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

-S|**s**

Disable / enable software flow control. The default depends on the mode: in raw mode (-E, the default), it's disabled; in edited mode (-e), it's enabled.

The order in which you specify the -E or -e, and -S or -s options matters:

Options	Mode	Software flow control
-e	Edited	Enabled
-S -e	Edited	Enabled
-e -S	Edited	Disabled
-E	Raw	Disabled
-s -E	Raw	Disabled
-E -s	Raw	Enabled

-v[v]...

Be verbose; additional v characters cause more verbosity.

The device-specific options that you can specify with the -d option are:

busno=*bus*

The bus number of the USB controller.

debug=[rx|tx|intr|all|none]

Copyright © 2014, QNX Software Systems Limited

Write raw URB data to *slogger* (p. 1807).

devno=*dev*

The USB address of the device.

did=*did*

The device ID of the device.

drt=*num*

Data ready timeout (device-specific).

ign_remove

Specify to prevent the removal callback from being attached.

module=*name*

Set the hardware module to use for an unknown vendor ID or device ID.

name=*name*

The basename for the pathname entry. The default is `/dev/serusb`.

path=*name*

Connect to the specified USB stack. Default is `/dev/io-usb/io-usb`.

pindex=*idx*

Set the port that the index options are to be applied to.

priority=*prio*

Set the priority of the event-processing thread. Default is 21.

query_modules

Display the currently supported hardware modules.

retry=*num*

The number of retries if a status of USBD_STATUS_DEV_NOANSWER is given.

rx_urbs=*num*

The number of URBs for BulkIn. Default is 4.

tx_urbs=*num*

The number of URBs for BulkOut. Default is 4.

Copyright © 2014, QNX Software Systems Limited

unit=*num*

The unit number: 1 = `/dev/serusb1`, 2 = `/dev/serusb2`, etc. Default is the first available number, starting from 1.

vid=*vid*

The vendor ID of the device.

wait=*num*

Wait *num* seconds for the USB stack. Default is 60 seconds.

Description:

The `devc-serusb` is a driver for USB-to-serial adaptors and Communications Device Class Abstract Control Model (CDC ACM) devices. The driver automatically detects ACM devices, based on the USB Class code.

If you provide the vid, did, busno and devno arguments for device-specific options, `devc-serusb` does not attach an insertion callback to detect newly inserted devices. The driver will work only with the already-inserted device that corresponds to the arguments specified on the command line. If you do not also specify the ign_remove option, the driver is terminated when the device is removed.

The unit option has meaning only when the driver is started for a specific device by using the vid, did, busno and devno arguments. If the driver is managing device insertions, the default behavior always applies.

Because `devc-serusb` is a USB class driver, the USB stack (`io-usb`) must be running before you start this driver.

This driver supports a `DCMD_CHR_RESET` *devctl()* command that resets the device. Not all drivers use this command.

Copyright © 2014, QNX Software Systems Limited

devc-serusb_dcd

Driver for USB-to-serial adaptors

You must be `root` to start this driver.

Syntax:

```
devc-serusb_dcd [options]
```

Runs on:

QNX Neutrino

Targets:

ARMLE-v7, x86

Options:

-b *baudrate*

The initial baud rate. The supported baud rates are 75, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600 (the default), 115200, 230400, and 460800.

-C *size*

The size of the canonical buffer, in bytes (default 256).

-d *args*[,*args* ...]

Device-specific options; see "*Device-specific options* (p. 303)," below.

-E

Set options to "raw" (default).

-e

Start in edited mode (default raw). Software flow control is enabled by default.

-F

Disable hardware flow control (default to hardware flow control enabled). Hardware flow control isn't supported in edited mode.

Copyright © 2014, QNX Software Systems Limited

-f

Enable hardware flow control (default). Hardware flow control isn't supported in edited mode.

-I ***number***

The size of the raw input buffer in bytes (default 2048).

-O ***number***

("Oh") The size of the output buffer, in bytes (default 1536).

-o ***opt*****[,*****opt*****]...**

("oh") A comma-separated list of additional options:

- nodaemon — don't run in the background.
- priority=*prio* — set the working priority of the internal pulse.

-Sls

Disable / enable software flow control. The default depends on the mode: in raw mode (-E, the default), it's disabled; in edited mode (-e), it's enabled.

The order in which you specify the -E or -e, and -S or -s options matters:

Options	Mode	Software flow control
-e	Edited	Enabled
-S -e	Edited	Enabled
-e -S	Edited	Disabled
-E	Raw	Disabled
-s -E	Raw	Disabled
-E -s	Raw	Enabled

-v[v]...

Be verbose; additional v characters cause more verbosity.

Device-specific options

The device-specific options that you can specify with the -d option are:

did=***device_ID***

The device ID of the serial device.

Copyright © 2014, QNX Software Systems Limited

iface_list=*number*[:*number*]...

Specify the USB interface number of the Communications class interface index. This option also causes the driver to immediately create a `/dev/serusb`*X* entry before the device is accessible to the host.

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb-dcd/io-usb`.

pindex=*num*

Set the port index for subsequent port-specific options.

priority=*prio*

Set the priority of the Rx, Tx, and USB stack events. The default priority is 21.

ubufsz=*size*

The USB transfer size. The default is the bulk endpoints' maximum packet size, typically 512; the buffer size is rounded up to the nearest multiple of 512.

vid=*vendor_ID*

The vendor ID of the serial device.

Description:

The `devc-serusb_dcd` is a driver for USB-to-serial adaptors, which in combination with the USB device stack (*`io-usb-dcd`* (p. 1018)), supports a CDC (Communications Device Class) interface presented to a connected USB host.

The *`io-usb-dcd`* (p. 1018) server must be running before you start this driver.

This driver creates a `/dev/serusb`*X* entry in the pathname space, where *X* is the next available integer, starting from 1. This happens when the device becomes accessible to the host, unless you specify the iface_list option, in which case it happens immediately.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start the driver with edited mode enabled, verbosity enabled, HW and SW flow control disabled, and a USB index of 0 (which also immediately reserves the `/dev/serusb`*X* path):

```
devc-serusb_dcd -e -v -F -s -d iface_list=0
```

Copyright © 2014, QNX Software Systems Limited

devc-serzscc

Zilog SCC serial communications manager (QNX Neutrino)

You must be `root` to start this driver.

Syntax:

```
devc-serzscc [[options]
              [port[^shift][+offset][,intr]]]... &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

The options are position-dependent and affect the subsequent ports.

-1

Enable only channel A for this device.

-2

Enable both channel A and B for this device.

-b *number*

The initial baud rate (default 57600).

-C *size*

The size of the canonical buffer in bytes (default 256).

-c *clock*[/*divisor*]

Define a custom clock rate, in hertz, and divisor for the serial port. The default is suitable for compatible serial ports.

-D *delay*

Inter-register access delay of *delay*.

-E

Copyright © 2014, QNX Software Systems Limited

Start in raw mode (the default). Software flow control is disabled by default.

-e

Start in edited mode (default raw). Software flow control is enabled by default.

-F

Disable hardware flow control (default to hardware flow control enabled). Hardware flow control is not supported in edited mode.

-f

Enable hardware flow control (default). Hardware flow control is not supported in edited mode.

-I ***number***

The size of the interrupt input buffer in bytes (default 2048).

-O ***number***

The size of the interrupt output buffer in bytes (default 2048).

-o ***opt*****[,*****opt*****...]**

Additional options, separated by commas. The options include:

- nodaemon — don't call *procmgr_daemon()* to make the driver run in the background. Use this option if you need to know when the device terminates.
- priority=*prio* — set the working priority of the internal pulse.

-SIs

Disable / enable software flow control. The default depends on the mode: in raw mode (-E, the default), it's disabled; in edited mode (-e), it's enabled.

The order in which you specify the -E or -e, and -S or -s options matters:

Options	Mode	Software flow control
-e	Edited	Enabled
-S -e	Edited	Enabled
-e -S	Edited	Disabled
-E	Raw	Disabled
-s -E	Raw	Disabled

Copyright © 2014, QNX Software Systems Limited

Options	Mode	Software flow control
-E -s	Raw	Enabled

-u *number*

Append *number* to the device name prefix (`/dev/ser`). The default is `1`; additional devices are given increasing numbers.

-v[v]...

Be verbose; additional `v` characters cause more verbosity.

port

Hex physical memory address of a serial port.

shift

The spacing of the registers as a power of 2. For example:

`0`

Registers are 1 byte apart.

`1`

Registers are 2 bytes apart.

`2`

Registers are 4 bytes apart.

...

n

Registers are 2^n bytes apart.

The default *shift* is `0`.

offset

Offset to add to the port value.

intr

Decimal interrupt used by this port.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devc-serzscc` manager is a small serial device manager for the QNX Neutrino RTOS. It supports the Zilog SCC chip.

All devices are fully interrupt driven and by default support standard hardware flow control on input and output (RTS/CTS). This can be disabled by the -F option.

Hardware flow control is not supported in edited mode.

If your application uses `/dev/console`, you should create a link from it to one of `/dev/ser1`, `/dev/ser2`, ... by adding a line like this to the buildfile used by `mkifs`:

```
[type=link] /dev/console = /dev/ser1
```

A read request by default returns when at least 1 character is available. To increase efficiency, you can control three parameters to control when a read is satisfied:

Time

Return after a specified amount of time has elapsed.

Min

Return when this number of characters are in the input buffer.

Char

Return if this forwarding character is in the input buffer.

If the Min value is greater than the size of the input buffer, the Min value is clipped to the size of the buffer. To avoid this, the size of the input buffer can be changed with the -I option.

These parameters are set using library routines (see *tcgetattr()*, *tcsetattr()*, *readcond()* and *TimerTimeout()* in the *C Library Reference*).

The `devc-serzscc` manager supports both raw and edited modes, making it a real tty device.

The following fields and flags are supported in the `termios` structure:

Field	Supported flags
c_cc	All characters
c_iflag	`BRKINT ICRNL IGNBRK IXON`

Copyright © 2014, QNX Software Systems Limited

Field	Supported flags
c_oflag	OPOST
c_cflag	CLOCAL CSIZE CSTOPB PARENB PARODD
c_lflag	ECHO ECHOE ECHOK ECHONL ICANON IEXTEN ISIG NOFLSH

Examples:

Start `devc-serzscc` in edited mode, specifying the clock rate, baud rate, and inter-register access delay:

```
devc-serzscc -e -c4915200/16 -b9600 -D4000 0x81000000^3+4,0x8002 &
```


Copyright © 2014, QNX Software Systems Limited

devf-generic

Generic flash filesystem

You must be `root` to start this driver.

Syntax:

```
devf-generic
    [-Aa] [-b priority] [-D] [-d log_method] [-E] [-e auto]
    [-f verifylevel] [-i arrayindex[,partindex]]
    [-L limit] [-l] [-m mountover]
    [-p backgroundpercent[,superlimit]] [-R] [-r]
    [-s base[,wsize[,aoffset[,asize[,usize[,bwidth[,ileave]]]]]]]
    [-t hi_water[,lo_water[,max]]] [-u update] [-V] [-v]
    [-w buffersize] [-x type]
```

Runs on:

QNX Neutrino

Targets:

Most flash devices

Options:

-A

When registering the path names for the partitions with *resmgr_attach()*, use the _RESMGR_FLAG_AFTER flag to force the path to be resolved after others with the same pathname at the same mountpoint.

-a

Don't automount filesystem partitions present on the media. If you specify both the -a and -R options, the driver ignores the -R option.

-b *priority*

Enable background reclaim at the specified *priority*. By default, background reclamation is disabled.

-D

Enable automatic detection of error-correcting code (ECC) mode. If you specify this option, you don't need to specify -x.

Copyright © 2014, QNX Software Systems Limited

Don't mix ECC-enabled partitions and ECC-disabled partitions; the driver doesn't support this.

-d *log_method*

Control the logging from the flash driver. The possible values for *log_method* are:

- 0 — log to *stdout* (the default).
- 1 — log to *slogger* (p. 1807).
- 2 — log to both *stdout* and `slogger`.

-E

Don't daemonize. If the driver detects a corrupt filesystem, the exit status is that filesystem's partition number plus 1.

-e *auto*

Only enumerate the flash partitions, instead of doing a full scan and `mount`. The flash driver automounts all partitions with a partition number less than or equal to *auto*.

For example, assume we have a flash layout as follows:

- `/dev/fs0p0` — raw
- `/dev/fs0p1` — formatted
- `/dev/fs0p2` — formatted
- `/dev/fs0p3` — formatted

If you start the driver with -e 1, the driver creates all the raw entries in `/dev`, but mounts only `/dev/fs0p1` (`/dev/fs0p0` is raw, so it's never mounted, regardless of the -e option)

-f *verifylevel*

Enable flash verify. (default=0, 0=none, write=1, erase=2, all=3)

-i *arrayindex*[,*partindex*]

Starting socket index and first partition index; 0 *index* 15. The default is 0,0. Use this to give multiple drivers unique IDs. The -i option is just a suggestion for the resource database manager; the selected indexes can be larger.

-L *limit*

Copyright © 2014, QNX Software Systems Limited

The number of retries to make if the physical flash erase function for a unit fails. The default is 0.

-l

List the available flash databases and exit.

-m ***mountover***

Override the mountpoints assigned to the file system that are formatted with an empty (i.e. `flashctl -p/dev/fs0p0 -f -n ""`) mountpoint. The *mountover* argument can include two %X format specifiers (like those for *printf()*) that are replaced by the socket index and the partition index.

The -m option doesn't override a mountpoint specified with *mkefs* (p. 1209).

-p ***backgroundpercent*[,*superlimit*]**

Set the background-reclaim percentage trigger (stale space over free space) and, optionally, the superseded extent limit before reclaim. The default is 100,16.

-R

Mount any automount filesystems as read-only. This option doesn't affect raw partition mounts, and it has an effect only at startup and initialization. Any subsequent mounting (with either *flashctl* (p. 771) or *mount* (p. 1312)) ignores the -R option. If you also specify the -a option, the driver ignores the -R option.

-r

Enable fault recovery for dirty extents, dangling extents, and partial reclaims.

You should always specify the -r option unless you're trying to debug an issue concerning flash corruption.

If you don't specify -r, and a power failure occurs, the following can happen:

- You can waste space. If an erasure was happening when the power was cut off, there will be some “dangling” extents (i.e. marked for deletion, but not actually deleted). If you specify the -vv option, the driver prints `dangle` for every dangling extent found. These extents will continue to occupy space forever, until they're deleted. Using the -r option will cause them to be reclaimed.

Copyright © 2014, QNX Software Systems Limited

- The system may be marked as read-only. If the driver detects an error in the structure of the filesystem, and you haven't specified the -r option, the driver marks the partition as read-only, so that it can't be further damaged.
- If a reclaim operation is interrupted by a power loss, the spare block may be unusable. In this case, if you specify the -vv option, the driver prints `partial` to the console. The partition is still read-write, but reclaims are turned off; if you continue to overwrite files, you'll eventually fill the filesystem with stale data.

-s *base*[,*wsize*[,*aoffset*[,*asize*[,*usize*[,*bwidth*[,*ileave*]]]]]]

Set socket options, normally the base physical address, window size, array offset, array size, unit size, bus width, and interleave. The format is left flexible for socket services with customized drivers. This option must be specified.

The arguments are:

base

Physical base address of the flash part. This value is board-specific.

wsize

Size of the physically contiguous flash part.

aoffset

For SRAM, the offset from the base address to the start of the flash array.

asize

For SRAM, the size of the flash array. The default is equal to *wsize*.

usize

The size of a physical erase sector. For SRAM, this number can be any power of two. 64 KB should be the minimum, for performance reasons.

bwidth

The total width of the data bus, as seen from the microprocessor's perspective. This is the width of one flash chip multiplied by the interleave. The value must be a power of 2 (1, 2, 4, or 8).

ileave

Copyright © 2014, QNX Software Systems Limited

The number of flash chips arranged on the data bus. Two 16-bit wide chips used as the upper and lower halves of a 32-bit databus give an interleave of 2. This number must be a power of 2 (1, 2, 4, or 8).

You can specify the base physical address, sizes, and offset in octal (`1000`), hexadecimal (`0x200`), or decimal (`512`). The sizes must be a power of two, and you can specify them with any of the following suffixes:

- (nothing) — bytes
- k — kilobytes
- m — megabytes

-t *hi_water*[,*lo_water*[,*max*]]

Set the high water, low water, and maximum attributes of the thread pool; the increment (i.e., the number of threads created at one time) is 1. The default is 4,2,100. The values must be related as follows:

- 0 < *hi_water* < *max*
- 0 *lo_water* *hi_water*
- *hi_water* < *max* 100

-u *update*

Specify the update level for timestamps. POSIX specifies that timestamps be kept when you access, create, or modify a file. FFSv3 is documented as not supporting the access timestamp, in order to reduce wear on the hardware.

The values for *update* are:

- 0 — don't update the modification time for files (the default).
- 1 — update the modification time for files according to the POSIX rules.
- 2 — update the modification time for files, as well as for the parent directory.

The -u2 option is very, very expensive and will cause many reclaims because the time updates have to flow right up to the root directory, so one file update may cause many directory updates.

-V

Display filesystem and MTD version information, and then exit.

-v

Copyright © 2014, QNX Software Systems Limited

Be verbose; specify additional v characters for more verbosity. For more information, see "*Verbose output* (p. 317)," below.

-w *buffersize*

Write (append) buffer size in bytes. The default *buffersize* is 512. Using a larger write-buffer prevents the creation of very small extents, reducing overhead. If *buffersize* is 0, appending is disabled.

-x *type*

Enable software ECC mode. Specify *type* as 1 for 64-byte alignment ECC, or 2 for 32-byte.

Don't mix ECC-enabled partitions and ECC-disabled partitions; the driver doesn't support this.

Description:

The `devf-generic` manager provides Flash filesystem support for any standard flash device. Typically, all you need to do is to pass the address and size using the -s option. The manager should detect the device automatically.

For information on creating a custom variant of `devf-generic` for your embedded system, see the Customizing the Flash Filesystem chapter of *Building Embedded Systems*

The default filenames are as follows (you can use the -i option to change the ID, *n*, appended to `/dev/fs`):

`/dev/fs`*n*

Default mountpoint for socket *n*.

`/dev/fs`*n*`p0`

Raw access for socket *n*, partition 0.

mountpoint

Flash filesystem mountpoint for socket *n*, partition 0 with transparent decompression.

You can specify the *mountpoint* above with the `mount` attribute of the *`mkefs`* (p. 1209) command, and override it with the -n option to *`flashctl`* (p. 771). By default, it's `/fs`*n*`p0`.

Copyright © 2014, QNX Software Systems Limited

If you erase a raw partition or the raw array (socket), you might erase any boot monitor, BIOS, or other data installed by the manufacturer. Check the documentation for the board.

The driver probes the hardware to determine its block size. If you need to know the block size, you can:

- Look in the documentation for the hardware.

  Or:

- Start the driver in verbose mode by specifying the -v option. In the output, `U:` indicates the number of units (also known as blocks or sectors), and `S:` indicates the block size. Both numbers are in hexadecimal.

  Or:

- Start the driver, and then run *`flashctl`* (p. 771), specifying the -i option.

The `devf-*` drivers support 32- and 64-byte error-correcting code (ECC) partitions. If you use 32-byte ECC:

- The interleave of the flash chips must be 1, or the driver reports an error.
- You must use the -b 5 option for `flashctl` to specify the alignment.
- You must use the `ecc_on=2` attribute in the buildfile for *`mkefs`* (p. 1209).

If you use 64-byte ECC:

- The interleave of the flash chips can be 1 or 2.
- You must use the -b 6 option for `flashctl` to specify the alignment.
- You must use the `ecc_on=1` attribute in the buildfile for *`mkefs`* (p. 1209).

The `mkefs` utility automatically handles the formatting alignment.

Verbose output

If you specify the -v option, a `devf-*` driver provides some useful information. This section describes the output that you get if you specify -vvv; at higher levels of verbosity, the output also includes messages about the use of *malloc()* and *free()*, but these aren't likely to be useful to you.

The output starts with something like this:

```
Flash Development Library - HEAD
Build: Jan 15 2010 13:00:00
MTD Build: Jan 15 2010, 13:25:55
```

These lines identify the source branch of the build, as well as the build times for `libfs-flash3.a` and `libmtd-flash.a`.

The rest of the messages that the driver prints are variable; messages are printed as the driver discovers things of interest. The general format of a message is

Copyright © 2014, QNX Software Systems Limited

`(devf tN::function:line)` *Message string*

where `tN` is the thread printing the message, *function* is the function emitting the message, and *line* is the line number that emitted the message.

The standard messages include the following:

`(devf t1::f3s_skt_attach:109) fs0 socket RAM (flash simulation)`

The flash driver located a flash socket, number zero (`fs0`). This message also indicates that the hardware identification succeeded.

`(devf t1::f3s_flash_probe:248) chip total = 1, bus_width = 8, interleave = 2`

After identifying the hardware, the driver prints the geometry. The chip total is the number of contiguous chips. The bus width is the size of the data bus, in bytes. The interleave is the number of physical chips sharing the data bus (high and low halves of the data bus, for example).

This particular example indicates that there are two physical chips sharing a 64-bit data bus.

`(devf t1::f3s_skt_attach:135) fs0 array SRAM U: 80 S: 020000`

The flash driver has allocated the flash array (i.e. the storage media) of type SRAM, with 0x80 hardware sectors, and a sector size of 0x020000 bytes.

`(devf t1::f3s_recover_boot:248) fs0p0 boot P[00] U: 80`

The filesystem is now scanning for partitions. It has found a boot header, and has named the partition `/dev/fs0p0`. The boot signature is located on physical block 0, and the partition has 0x80 sectors (called “units” in `devf-*` nomenclature).

`(devf t1::f3s_recover_reclaim:989) fs0p0 spare P[7F]`

The second phase of the startup scan is processing `/dev/fs0p0`, and has found a spare block. The spare block is located on physical sector 0x7F.

`(devf t3::f3s_table_find:66) fs0p1 raw U: 09`

A region of flash was found that isn't formatted. The region is given the name `/dev/fs0p1`; its size is 0x09 sectors.

Examples:

Start `devf-generic` and automatically mount the flash filesystem partitions at the base address 0xFF000 with a window size of 16 megabytes, with an initial fault

Copyright © 2014, QNX Software Systems Limited

recovery process, most POSIX semantics enabled and background reclaim at priority 5:

```
devf-generic -s 0xFF000,16M -r -u2 -b5 &
```

Create a 32 MB flash partition, with a 64 KB unit (sector) size:

```
devf-generic -s0,32m,,,64k  -v -r
```

Create a 128 MB flash partition, with large block sizes (to speed formatting):

```
devf-generic -s0,128m,,,512k -v -r
```

Create a 4 MB partition:

```
devf-generic -s0,4m,,,64k -v -r
```

Create a 16 MB flash partition, from a given physical address, with a 128 KB unit size, and a 16-bit wide data bus:

```
devf-generic -s0xa4000000,16m,,,128k,2 -v -r
```

Create a 16 MB flash partition, from a given physical address, with a 256 KB unit size, and a 32-bit-wide data bus, with an interleave of two:

```
devf-generic -s0,16m,,,256k,4,2 -v -r
```

Caveats:

You must specify the -s option when using this driver.

Although the Flash filesystem supports most POSIX semantics, some functionality isn't implemented in order to keep the driver simple and efficient. The unsupported POSIX semantics include:

- Hard links, and everything related to hard links (the `.` and `..` directories don't exist, `struct stat`'s *nlink* member is hard-coded, and *unlink()* of directories returns ENOTSUP).
- Access times aren't updated on the media; they're set to the modification time.

QNX Neutrino flash filesystem version 3 no longer provides built-in decompression. The flash filesystem's decompression functionality has moved into the ***inflator*** (p. 984) resource manager. You should now use the ***deflate*** (p. 183) utility to compress files.

Performance might be slow when multiple writers are writing randomly to a shared file or to a shared directory (e.g. using `unlink` or `rename`). In these cases, the offset pointers have to be rewound for every access. There's no performance penalty when appending to a file, or when creating files with *open*(`O_CREAT`), `mkdir`, `mknod`, or `link`.

Copyright © 2014, QNX Software Systems Limited

devf-ram

Simulate flash filesystem using RAM memory

You must be `root` to start this driver.

Syntax:

```
devf-ram
    [-a] [-b priority]
    [-E] [-f verifylevel] [-i arrayindex[,partindex]]
    [-l] [-m mountover]
    [-p backgroundpercent[,superlimit]] [-R] [-r]
    [-s
base[,wsize[,aoffset[,asize[,usize[,bwidth[,ileave]]]]]]]
    [-t threads] [-u update] [-V] [-v]
    [-w buffersize]
```

Runs on:

QNX Neutrino

Targets:

ARMv7 x86

Options:

-a

Don't automount filesystem partitions present on the media. If you specify both the -a and -R options, the driver ignores the -R option.

-b *priority*

Enable background reclaim at the specified *priority*. By default, background reclamation is disabled.

-E

Don't daemonize. If the driver detects a corrupt filesystem, the exit status is that filesystem's partition number plus 1.

-f *verifylevel*

Simulate flash verify; only provided for syntax compatibility with real flash hardware (default=0, 0=none, write=1, erase=2, all=3).

Copyright © 2014, QNX Software Systems Limited

-i ***arrayindex*[,*partindex*]**

Starting socket index and first partition index; 0 *index* 15. The default is 0,0. Use this to give multiple drivers unique IDs. The -i option is just a suggestion for the resource database manager; the selected indexes can be larger.

-l

List the available flash databases and exit.

-m ***mountover***

Override the mountpoints assigned to a file system that are formatted with an empty (i.e. `flashctl -p/dev/fs0p0 -e -f -n ""`) mountpoint. The *mountover* argument can include two %X format specifiers (like those for *printf()*) that are replaced by the socket index and the partition index.

The -m option doesn't override a mountpoint specified with *mkefs* (p. 1209).

-p ***backgroundpercent*[,*superlimit*]**

Set the background-reclaim percentage trigger (stale space over free space) and, optionally, the superseded extent limit before reclaim. The default is 100,16.

-R

Mount any automount filesystems as read-only. This option doesn't affect raw partition mounts, and it has an effect only at startup and initialization. Any subsequent mounting (with either *flashctl* (p. 771) or *mount* (p. 1312)) ignores the -R option. If you also specify the -a option, the driver ignores the -R option.

-r

Enable fault recovery for dirty extents, dangling extents, and partial reclaims.

You should always specify the -r option unless you're trying to debug an issue concerning flash corruption.

If you don't specify -r, and a power failure occurs, the following can happen:

- You can waste space. If an erasure was happening when the power was cut off, there will be some “dangling” extents (i.e. marked for deletion, but not actually deleted). If you specify the -vv option, the driver prints `dangle` for every dangling extent found. These extents will continue to

Copyright © 2014, QNX Software Systems Limited

occupy space forever, until they're deleted. Using the -r option will cause them to be reclaimed.

- The system may be marked as read-only. If the driver detects an error in the structure of the filesystem, and you haven't specified the -r option, the driver marks the partition as read-only, so that it can't be further damaged.
- If a reclaim operation is interrupted by a power loss, the spare block may be unusable. In this case, if you specify the -vv option, the driver prints `partial` to the console. The partition is still read-write, but reclaims are turned off; if you continue to overwrite files, you'll eventually fill the filesystem with stale data.

-s ***base*****[,***wsize*****[,***aoffset*****[,***asize*****[,***usize*****[,***bwidth*****[,***ileave*****]]]]]]**

Set socket options, normally the base physical address, window size, array offset, array size, unit size, bus width, and interleave. The format is left flexible for socket services with customized drivers.

The arguments are:

base

Physical base address of the flash part. This value is board-specific.

For the `devf-ram` utility, the *base* argument carries a special meaning:

0

Allocate system memory.

Nonzero

Use the exact physical address. You must exercise caution here. See the caveats below.

---

wsize

Size of the physically contiguous flash part.

aoffset

For SRAM, the offset from the base address to the start of the flash array.

asize

Copyright © 2014, QNX Software Systems Limited

For SRAM, the size of the flash array. The default is equal to *wsize*.

usize

The size of a physical erase sector. For SRAM, this number can be any power of two. 64 KB should be the minimum, for performance reasons.

bwidth

The total width of the data bus, as seen from the microprocessor's perspective. This is the width of one simulated flash chip multiplied by the interleave. This value must be a power of 2 (1, 2, 4, or 8).

ileave

The number of simulated flash chips arranged on the data bus. This value must be a power of 2 (1, 2, 4, or 8).

You can specify the base physical address, sizes, and offset in octal (`0777`), hexadecimal (`0x1ff`), or decimal (`511`). The sizes must be a power of two, and you can specify them with any of the following suffixes:

- (nothing) — bytes
- k — kilobytes
- m — megabytes

On ARM targets, `devf-ram` can't resize the shared object `/dev/shmem/fs*`. If you need to restart `devf-ram` with a new size, first unlink the old shared object:

```
rm /dev/shmem/fs*
```

-t ***threads***

The number of threads. The minimum is 1, the default is 2, and the maximum is 100. Extra threads increase performance when background reclaiming is enabled (with the -b option) and when multiple chips and/or spare blocks are available.

-u ***update***

Specify the update level for timestamps. POSIX specifies that timestamps be kept when you access, create, or modify a file. FFSv3 is documented as

Copyright © 2014, QNX Software Systems Limited

not supporting the access timestamp, in order to reduce wear on the hardware.

The values for *update* are:

- 0 — don't update the modification time for files (the default).
- 1 — update the modification time for files according to the POSIX rules.
- 2 — update the modification time for files, as well as for the parent directory.

The -u2 option is very, very expensive and will cause many reclaims because the time updates have to flow right up to the root directory, so one file update may cause many directory updates.

-V

Display filesystem and MTD version information, and then exit.

-v

Be verbose; specify additional v characters for more verbosity. For more information, see "*Verbose output* (p. 317)" in the entry for *devf-generic* (p. 311).

-w *buffersize*

Write (append) buffer size in bytes. The default *buffersize* is 512. Using a larger write-buffer prevents the creation of very small extents, reducing overhead. If *buffersize* is 0, appending is disabled.

Description:

The `devf-ram` manager simulates flash filesystem in RAM using the following default filenames (the ID, *n*, appended to `/dev/fs` can be changed via the -i option):

`/dev/fs`*n*

Default mountpoint for socket *n*.

`/dev/fs`*n*`p0`

Raw access for socket *n*, partition 0.

mountpoint

Flash filesystem mountpoint for socket *n*, partition 0 with transparent decompression.

Copyright © 2014, QNX Software Systems Limited

You can specify the *mountpoint* above with the `mount` attribute of the *mkefs* (p. 1209) command, and override it with the -n option to *flashctl* (p. 771). By default, it's `/fs`*n*`p0`.

Examples:

Start `devf-ram` with a 16 MB partition.

```
devf-ram  -s0,16m
```

Start `devf-ram` and automatically mount the flash filesystem partitions, with an initial fault recovery process, most POSIX semantics enabled and background reclaim at priority 5 (default size: 1 MB):

```
devf-ram -r -u2 -b5 &
```

Create a 32 MB flash partition, allocated from system RAM, with a 64 KB unit (sector) size:

```
devf-ram -s0,32m,,,64k  -v -r
```

Create a 128 MB flash partition in system RAM, with large block sizes (to speed formatting):

```
devf-ram -s0,128m,,,512k -v -r
```

Create a 4 MB partition from system RAM:

```
devf-ram -s0,4m,,,64k -v -r
```

You must format and erase a `devf-ram` partition before you can mount the flash filesystem. See the caveats below.

If you specify a block size in your buildfile for DRAM-based flash filesystems, limit the size to the default, which is 64 KB.

Caveats:

Although the flash filesystem supports most POSIX semantics, some functionality isn't implemented in order to keep the driver simple and efficient. The unsupported POSIX semantics include:

- Hard links, and everything related to hard links (the `.` and `..` directories don't exist, `struct stat`'s *nlink* member is hard-coded, and *unlink()* of directories returns ENOTSUP).
- Access times aren't updated on the media; they're set to the modification time.

QNX Neutrino flash filesystem version 3 no longer provides built-in decompression. The flash filesystem's decompression functionality has moved into the *inflator* (p.

Copyright © 2014, QNX Software Systems Limited

984) resource manager. You should now use the *deflate* (p. 183) utility to compress files.

Performance might be slow when multiple writers are writing randomly to a shared file or to a shared directory (e.g. using `unlink` or `rename`). In these cases, the offset pointers have to be rewound for every access. There's no performance penalty when appending to a file, or when creating files with *open*(`O_CREAT`), `mkdir`, `mknod`, or `link`.

Don't try to create a `devf-ram` partition at the address of a real flash memory. You may get an error message: `Unable to properly identify any flash devices`.

Don't try to create a `devf-ram` partition (e.g. using nonzero value for *base* argument) at the address of physical memory that is in use. It may destroy applications and crash the operating system. The only use for specifying such nonzero *base* is to create a flash filesystem for board specific memory (e.g. SRAM).

You must format and erase a `devf-ram` partition before you can mount the flash filesystem. e.g.

```
devf-ram -s0,16m
flashctl -p /dev/fs0p0 -e -f -m
```

If there's insufficient RAM, when you try to create an *n*M size partition with -s0 option, the `devf-ram` driver returns without an error message. The partition isn't created.

Copyright © 2014, QNX Software Systems Limited

devh-egalax.so

Driver for USB Egalax touch devices

Syntax:

```
io-hid -d egalax [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

device=0x*XXXX*

Pass in an alternative device ID.

info

Gather more information on the controller. This is only for general information purposes and simply sends the information to *slogger* (p. 1807).

noinit

Don't initialize the controller.

Don't use this option unless you're certain that the default protocol your controller is using is the Egalax protocol.

upath=*path*

The USB stack path; the default is `/dev/io-usb/io-usb`.

vendor=0x*XXXX*

Pass in an alternative vendor ID.

verbose=*level*

Be verbose and specify the level of debug information (range 1-5).

wait=*num*

Copyright © 2014, QNX Software Systems Limited

The number of seconds to wait for the USB stack to come up.

Description:

The Egalax touchscreens aren't HID-compliant, so the `devh-egalax.so` DLL converts Egalax native controller packets into generic HID packets, which *`devi-hid`* (p. 339) then handles.

If you're using this driver with the USB (*`devh-usb.so`* (p. 337)) module, you must specify the igndev option to `devh-usb.so`, specifying the Egalax vendor and device IDs.

Examples:

Start `io-hid` using the Egalax driver, and then start `devi-hid`:

```
io-hid -d egalax &
devi-hid touch
```

Start `io-hid` using the Egalax driver at high verbosity and with a new stack path:

```
io-hid -d egalax verbose=5,upath=/dev/payton-usb/ &
```

Files:

`devh-egalax.so`

The `devh-egalax.so` DLL is normally found in `/lib/dll`.

Copyright © 2014, QNX Software Systems Limited

devh-microtouch.so

Driver for Microtouch EXII USB touch devices

Syntax:

```
io-hid -d microtouch [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=0x*XXXX*

The device ID.

noscaling

Disable coordinate scaling. The coordinates are normally scaled down to 14-bit values, due to an issue with the upper layers of the input driver sharing a data type that uses `signed short` values.

Since the touchscreen has a resolution of 16 bits, a `short` isn't big enough to handle the data correctly. If the client is the resource manager interface (composition manager) and *not* the Advanced Graphics system, then you can specify the noscaling option as it bypasses the `short` values.

upath=*path*

The USB stack path; the default is `/dev/io-usb/io-usb`.

verbose=*level*

Be verbose and specify the level of debug information (range 1-5).

vid=0x*XXXX*

The vendor ID.

wait=*num*

The number of seconds to wait for the USB stack to come up.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devh-microtouch.so` DLL converts Microtouch EXII packets into generic HID packets, which *`devi-hid`* (p. 339) then handles.

Examples:

Start `io-hid` using the Microtouch driver, and then start `devi-hid`:

```
io-hid -d microtouch &
devi-hid touch
```

Start `io-hid` using the Microtouch driver at high verbosity and with a new stack path and a new device ID:

```
io-hid -d microtouch did=0x1234,verbose=5,upath=/dev/huxley-usb/ &
```

Files:

`devh-microtouch.so`

The `devh-microtouch.so` DLL is normally found in `/lib/dll`.

Copyright © 2014, QNX Software Systems Limited

devh-ps2ser.so

Driver for serial and PS2 human interface devices (HID)

Syntax:

```
io-hid -d ps2ser
       protocol_module[,options]:device_module[,options]
       [:protocol_module[,options]:device_module[,options]]...

       [opts=v[v...]]
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Colons (`:`) separate modules; commas (`,`) separate module options.

opts=v[v...]

Increment verbosity. The default is zero verbosity.

***protocol_module*[,*options*]**

The input protocol module. You can specify multiple protocol/device module pairs.

The *protocol_module* variable may be any one of the following modules:

ps2mouse

Regular PS/2 mouse. There are no options.

msoft[,*options*]...

Mouse compatible with Microsoft/IntelliMouse protocol (serial). Options:

- b=*baud* — Baud rate for serial device (default 1200)
- 3 — Microsoft 3 button mouse
- R — Don't reset mouse (the default is to reset the mouse)

Copyright © 2014, QNX Software Systems Limited

- i — IntelliMouse protocol (wheel mice)

msys[,*options*]...

Mouse compatible with Mouse Systems mouse protocol (used by Logitech). Options:

- b=*baud* — Baud rate for serial device (default 1200)
- R — Don't reset mouse (default reset mouse)

kbd[,*options*]...

Scan codes for primary keyboard. Options:

- k=(*rate*[,*delay*]) — Keyboard rate (Hz), delay (ms). If you continually depress a key, after *delay* milliseconds, it will input data *rate* times a second. The default is 30Hz after 500ms. This only works in conjunction with the kbddev device module.
- R — Don't reset the device while doing a protocol reset
- r — Reset the device while doing a protocol reset
- s — The device driver should supply valid symbols

***device_module*[,options]**

Device modules. The *device_module* variable depends on the *protocol_module* specified. The following device modules are supported:

fd[,*options*]...

Opens a device via the *open()* function. Options:

- d=*device* — The device to open *fd()* on (default `/dev/ser1`)
- s — Specifies that the input interface is serial (this allows the module to use some `devctl` commands that are specific to a serial port)
- P — The processing priority for this input event

uart[,*options*]...

Enables direct access to `8250`, `16450`, and `16550` uarts. Options:

- i=*irq* — IRQ number for the serial device (default 4)
- p=*ioport* — The port of the serial device (default `3f8`)
- 1 — Use COM1
- 2 — Use COM2

Copyright © 2014, QNX Software Systems Limited

- P — The processing priority for this input event

kbddev[,*options*]...

For PS2 keyboard. Options:

- f=*filename* — Create the named file and collect all data passed to the protocol level (debug only)
- i=*irq* — IRQ number (default 1)
- p=(*ioport*,*add*) — Port address (default `0x60`), and a value to add to get the status port address (default 4: `0x60` + 4 = `0x64`)
- r — Reset the device on initiation
- P *priority* — The processing priority for keyboard events

mousedev[,*options*]...

For PS2 mouse. Options:

- f=*filename* — Create the named file and collect all data passed to the protocol level (debug only)
- i=*irq* — IRQ number (default 12)
- p=(*ioport*,*add*) — Port address (default `0x60`), and a value to add to get the status port address (default 4: `0x60` + 4 = `0x64`)
- r — Reset the device on initiation
- P *priority* — The processing priority for mouse events

Description:

The `devh-ps2ser.so` DLL provides `io-hid` with HID information. The DLL collects raw data from a variety of legacy devices (e.g., PS2 keyboards, and PS2 and serial mice), transforms it to unified USB HID report format, and then sends the data to `io-hid`. The `io-hid` manager forwards the data to `devi-hid`.

- The `devh-ps2ser.so` DLL is the low-level part of the input channel. You have to start *`devh-usb.so`* (p. 337) to provide any top-level services required.
- If you press several keys at once on some Microsoft keyboards, the keyboard doesn't produce any indication when you release the keys. As a result, the input driver thinks you're still holding the keys down. For more information, see *`http://support.microsoft.com/kb/909528/en-us`*.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start a regular PS/2 mouse, a Microsoft/IntelliMouse serial mouse on COM1, and a PS/2 keyboard:

```
io-hid -d ps2ser ps2mouse:mousedev:msoft:uart,1:kbd:kbddev
```

Files:

devh-ps2ser.so

The devh-ps2ser.so Dll is normally found in /lib/dll.

Errors:

If an error occurs in devh-ps2ser.so, the keyboard will not work in text mode. If you specify at least one v option, details of driver activity will be reported on the console screen and will be appended to the system log; for more detail, increase the verbosity level.

Copyright © 2014, QNX Software Systems Limited

devh-touchintl.so

Driver for USB Touch International touch devices

Syntax:

```
io-hid -d touchintl [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=0x*XXXX*

The device ID.

noinit

Don't initialize the controller.

rate=*rate*

The coordinate output rate, in the range 1–7:

- 1 — kiosk mode; data is sent only on a touch, not on a drag.
- 2 — 30 pps (points per second)
- 3 — 50 pps
- 4 — 80 pps (the default value)
- 5 — 100 pps
- 6 — 130 pps
- 7 — 150 pps

upath=*path*

The USB stack path; the default is `/dev/io-usb/io-usb`.

verbose=*level*

Be verbose and specify the level of debug information (range 1-5).

vid=0x*XXXX*

Copyright © 2014, QNX Software Systems Limited

The vendor ID.

wait=*num*

The number of seconds to wait for the USB stack to come up.

Description:

The Touch International touchscreens aren't HID-compliant, so the `devh-touchintl.so` DLL converts touchintl native controller packets into generic HID packets, which *`devi-hid`* (p. 339) then handles.

Examples:

Start `io-hid` using the Touch International driver, and then start `devi-hid`:

```
io-hid -d touchintl &
devi-hid touch
```

Start `io-hid` using the Touch International driver at high verbosity and with a new stack path and a new device ID, at a data rate of 100 packets per second:

```
io-hid -d touchintl did=0x1234,rate=4,verbose=5,upath=/dev/payton-usb/ &
```

Files:

`devh-touchintl.so`

The `devh-touchintl.so` DLL is normally found in `/lib/dll`.

Copyright © 2014, QNX Software Systems Limited

devh-usb.so

Driver for USB-compliant human interface devices (HID)

Syntax:

```
io-hid -d usb [option[,option ...]] ... &
io-hid -d usb [option[,option ...]] ...
```

Runs on:

QNX Neutrino

Options:

igndev=*vid*[:*did*]

Don't attach to the HID device matching the given vendor and device IDs. The *vid* and *did* must be hexadecimal vendor and product IDs (e.g. igndev=0x1234:0x5678), as reported by the *usb* (p. 2025) utility.

You can use the igndev option to ignore specific USB HID devices. This allows another USB driver to attach and manage the device. You can specify multiple igndev options.

The *did* also supports pattern matching to let you specify a range for device IDs for a particular vendor; see the examples below.

priority=*prio*

Run the removal thread at priority *prio*. The default priority is 8.

upath

The USB stack path. The default is `/dev/io-usb/io-usb`

verbose

Be verbose.

wait=*num*

The number of seconds to wait for the USB stack to come up.

Description:

The `devh-usb.so` is a DLL which is used with the `io-hid` manager. It connects to the USB stack to give HID clients access to USB-compliant human interface devices.

Copyright © 2014, QNX Software Systems Limited

Run the USB stack first. Please refer to *io-usb* (p. 1015) for further information.

Examples:

Start `io-hid` using the USB HID driver and manage all USB HID devices:

```
io-hid -d usb &
```

or:

```
io-hid &
mount -Tio-hid devh-usb.so
```

Ignore all devices for a specific vendor:

```
io-hid -dusb igndev=0x1234
```

Ignore a specific device for a specific vendor:

```
io-hid -dusb igndev=0x1234:0x5678
```

Ignore a range of devices for a specific vendor:

```
io-hid -dusb igndev=0x1234:0x5???
```


Copyright © 2014, QNX Software Systems Limited

devi-hid

HID input manager

Syntax:

```
devi-hid [general_opts]
         protocol* [protocol_opts]*
         filter* [filter_opts]*
```

Runs on:

QNX Neutrino

Targets:

Any supported platform that has `io-hid` running.

Options:

General options

-l

List the internal modules. Modules are listed in the following format where `class` may be `P` (protocol) or `F` (filter):

```
module name | date last compiled | revision | class
```

-R *res_x*,*res_y*

Specify the display resolution (e.g., -R800,480).

-U *user_name*

-U *uid*[:*gid*[,*sup_gid*]*]]

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

-v[v]...

Copyright © 2014, QNX Software Systems Limited

Verbose output. More **v** characters cause more verbosity.

Protocol modules

The protocol modules and their options are:

kbd

Keyboard scan codes (connected to primary keyboard). The options include:

-k ***rate*****[,*****delay*****]**

The keyboard rate (Hz),delay (ms). If you continually depress a key, after *delay* milliseconds, it will input data *rate* times a second. The default is 150Hz after 500ms.

-u ***device***

Optional; USB device number.

mouse

Common mouse protocol (no options).

touch

HID-compliant touchscreen. The options include:

-K

Use kiosk mode (cursor doesn't drag).

-m ***size***

The touchscreen Matrix size (default 1023). This is used for calculated transformation points.

-u ***device***

Optional; USB device number.

Filter modules

The filter modules and their options are:

keyboard

Translate scan codes to Unicode. The options include:

-k ***kbd_file***

Copyright © 2014, QNX Software Systems Limited

The file to use to map the keyboard to support international languages or alternate layouts, such as Dvorak.

-L [N][C][S]

Set the initial state of the keyboard and its LEDs (by default, all are off):

- C — turn CapsLock on
- N — turn NumLock on
- S — turn ScrollLock on

rel

Filter and compress the relative coordinates of mouse events. The options include:

-a

Wheel acceleration parameter (default 10); the higher this value, the faster the mouse wheel acceleration.

-G *gain*

Motion multiplier (default 1).

-I

Swap right and left buttons.

-T *threshold*

Speed-doubling threshold in mickeys/second. If the mouse speed exceeds this threshold, the cursor will move twice as far as it normally does per mickey. (A mickey is the smallest amount of motion the mouse can detect.) The default threshold is 100.

-x

Reverse X.

-y

Reverse Y.

abs

Transform and compress absolute coordinate “touch” events. The options include:

Copyright © 2014, QNX Software Systems Limited

-a 3|4

The transformation algorithm to use. The available algorithms are 3-point (the default) and 4-point.

-b

Touching the screen is a right mouse button (default left).

-c

Calibrate mode; don't transform coordinates.

-D *pixels*

Use a differential filter that filters out packets that are within the given number of pixels of the last touch. This prevents sending repeat coordinates.

-f *filename*

The name of the calibration file; the default is `/etc/system/config/calib.`*hostname*.

-N *pixels*

Use a noise filter that filters out packets that are outside the given maximum number of pixels. This filters out jittery events from some touchscreens.

-O *origin*

("Oh") The touchscreen origin, (0,0). This is used for the calibration-free transformation. If no calibration file is specified, the driver attempts to calculate the X,Y position based on the screen resolution and the touchscreen matrix size (if known).

The possible values are:

- 0 — upper left
- 1 — lower left
- 2 — lower right
- 3 — upper right

-o *x,y*

The origin of the display region; the default is the origin of the graphics region.

-S *ms*

Copyright © 2014, QNX Software Systems Limited

Use soft button release, which emulates a release event after the given number of milliseconds of inactivity from the touch controller. This is helpful for screens that aren't very sensitive, so that every event received before the given number of milliseconds expire is treated as a pen-down event.

-s *x,y*

The coordinates of the lower right corner of the display region; The default is the width and height of the graphics region.

-x

Reverse X.

-y

Reverse Y .

Description:

This manager is a universal input daemon for human interface devices (HIDs). It's a client of `io-hid`, the HID server.

The *`io-hid`* (p. 1005) resource manager must be running before `devi-hid` can start.

This manager doesn't need information about the physical interfaces of real devices: it relies on service from the `io-hid` resource manager and supplementary input modules.

If you specify the verbosity option, activity messages are sent to the console screen and to the system log.

Examples:

Start the keyboard and mouse manager:

```
devi-hid kbd mouse
```

Start the HID driver keyboard and touchscreen and tell the touchscreen to use the older, four-point calibration algorithm:

```
devi-hid kbd touch abs -a4
```

Files:

`libhiddi.so`

Copyright © 2014, QNX Software Systems Limited

Used by `devi-hid`

Errors:

If an error occurs in starting `devi-hid`, the HID won't work. If you specify at least one v option, activity details will be reported on the console screen and will be appended to the system log; for more detail, increase the verbosity level.

Copyright © 2014, QNX Software Systems Limited

devn-crys8900.so

Driver for Crystal 89xx Ethernet adapters

Syntax:

```
io-pkt-variant -d crys8900 [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0

BNC

1

UTP

2

AUI

3

FIBER

dma=*num*

The DMA channel.

duplex=*0|1*

Copyright © 2014, QNX Software Systems Limited

Half (0) or full (1) duplex mode.

iftype=*num*

Interface type (from <net/if_types.h>). The default is IFT_ETHER.

iorange=*0xXXXXXXXX*

I/O base address.

irq=*num*

IRQ of the interface. The default is automatically detected on supported hardware (but see caution below).

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of the controller. The default is automatically detected on supported hardware.

media=*num*

Media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

priority=*N*

Priority of the driver event-thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

uptype=*name*

Interface name. The default is “en”.

Copyright © 2014, QNX Software Systems Limited

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

Description:

The devn-crys8900.so driver controls Crystal 89xx Ethernet adapters. This is a legacy io-net driver; its interface names are in the form en*X*, where *X* is an integer.

This driver can't always detect the correct irq and ioport options, especially for ISA devices. To be sure, always specify irq and ioport when using this driver.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- ip4csum, ip4csum-rx, ip4csum-tx
- tcp4csum, tcp4csum-rx, tcp4csum-tx
- tcp6csum, tcp6csum-rx, tcp6csum-tx
- udp4csum, udp4csum-rx, udp4csum-tx
- udp6csum, udp6csum-rx, udp6csum-tx

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start io-pkt-v4-hc using the Crystal 89xx driver: stack:

```
io-pkt-v4-hc -d crys8900
ifconfig en0 10.1.0.184
```

Files:

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-dm9102.so

Driver for Davicom DM9102 Ethernet adapters

Syntax:

```
io-pkt-variant -d dm9102 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options. These options will override auto-detected defaults.

did=*0xXXXX*

Device ID.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of the controller. If no SROM is available, the MAC will default to `00:00:00:00:00:00`

nomulticast

Disable multicast support.

pci=*0xXXXX*

PCI index of the controller.

phyaddr=*num*

Copyright © 2014, QNX Software Systems Limited

Override the mii routines and use the specified phy address.

pktque=*num*

Limit the number of packets in the queue. The default is 100.

priority

Priority of the driver thread. The default is 21.

promiscuous

Enable promiscuous mode.

receive=*num*

Set the number of receive descriptors. The default is 64.

single

Configure and run only the first DM9102 card that is found (single instance).

speed=10|100

Media data rate (10 Mbit or 100 Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

threshold=*N*

The amount of packet data that must be in the TX FIFO before transmission is initiated. The range is 0–4. The default is 3.

transmit=*num*

Set the number of transmit descriptors. The default is 128.

verbose

Be verbose.

vid=*0xXXXX*

PCI vendor ID.

Description:

The `devn-dm9102` driver controls Davicom DM9102 Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

Copyright © 2014, QNX Software Systems Limited

`ifconfig en`*X*

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the DM9102 driver:

```
io-pkt-v4-hc -d dm9102
ifconfig en0 10.0.0.184
```

Files:

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-el509.so

Driver for 3Com 509 ISA Ethernet adapters

Syntax:

```
io-pkt-variant -d el509 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*type*

Network cable connector type:

0

BNC

1

UTP

2

AUI

3

FIBER

The default is automatically detected on supported hardware.

ioport=*port*

Copyright © 2014, QNX Software Systems Limited

The I/O port of the interface. The *port* parameter must be a hex address (e.g. 0x320). The default is automatically detected on supported hardware (but see caution below).

irq=*req*

The IRQ of the interface. The default is automatically detected on supported hardware (but see caution below).

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

mtu=*X*

Maximum transmission unit. The default is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pcmcia

Flag to indicate a PCMCIA device to the driver.

promiscuous

Enable promiscuous mode. The default is off.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

Description:

The `devn-el509.so` driver controls 3Com 509 ISA Ethernet adapters.

This driver can't always detect the correct irq and ioport options, especially for ISA devices. To be sure, always specify irq and ioport when using this driver.

You can use *ifconfig* (p. 957) to enable hardware checksums if supported.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start `io-pkt-v6-hc` using the 509 ISA Ethernet adapter driver:

```
io-pkt-v6-hc -d el509 ioport=0x320,irq=11
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-el900.so

Driver for 3Com 90x Network Interface Cards

Syntax:

```
io-pkt-variant -d el900 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0
BNC

1
UTP

2
AUI

3
FIBER

The default is automatically detected on supported hardware.

did=*0xXXXX*

PCI device ID. The default is automatically detected on supported hardware.

duplex=*0|1*

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex

Copyright © 2014, QNX Software Systems Limited

alone is specified, it is ignored and both speed and duplex are auto-negotiated.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 64.

transmit=*num*

Set the number of transmit descriptors. The default is 128.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

PCI vendor ID. The default is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devn-el900.so` driver controls 3Com 90x Network Interface Cards (NICs). The IRQ of the interface is automatically detected on supported hardware. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v6-hc` using the 90x NIC driver:

```
io-pkt-v6-hc -d el900 verbose
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-epic.so

Driver for SMC 9432 (EPIC) Ethernet adapters

Syntax:

```
io-pkt-variant -d epic [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|3*

Network cable connector type:

0

BNC

1

UTP

3

FIBER

did=*0xXXXX*

Device ID. The default is automatically detected on supported hardware.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

Copyright © 2014, QNX Software Systems Limited

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

mmap

Use memory mapped registers. The default is IO mapped.

mru=*X*

Maximum receive unit. The default is 1514.

mtu=*X*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

priority

Priority of the driver thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 64.

single

Use this option if you have multiple Epic cards in your system and want to configure them differently. The single option tells the driver to stop after configuring the first Epic card it finds. The order of the search can't be determined because it depends on the PCI server and PCI BIOS used. After the first card has been configured, when the driver is invoked again, it will find and configure the next card in order, and so on until all Epic cards have been configured. The default is to enable all Epic cards found.

Copyright © 2014, QNX Software Systems Limited

speed=10|100

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

transmit=*num*

Set the number of transmit descriptors. The default is 128.

verbose

Be verbose. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-epic.so` driver controls SMC 9432 (EPIC) Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

`ifconfig en`*X*

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v6-hc` using the EPIC driver:

```
io-pkt-v6-hc -d epic verbose,speed=10,duplex=0
ifconfig en0 10.1.0.184
```

Files:

Copyright © 2014, QNX Software Systems Limited

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-fd.so

File descriptor interface driver

Syntax:

```
io-pkt-variant -d fd fd=device[,option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

ahdlc

Read or write packet data in AHDLC frame format.

fd=*device*

The device on which to open the file descriptor to read or write packet data. You must specify this option.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

The MAC address of the controller. There is no default.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

nomulticast

Disable multicast support.

priority=*N*

Priority of the driver event-thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

Description:

The `devn-fd.so` driver uses file-descriptor based I/O (i.e. *open()*, *read()*, *write()*, and so on) to receive and transmit packets. It provides the Network Manager (*io-pkt** (p. 1007)) with reliable data transfer over any media supported by a file-descriptor-based server process.

The `devn-fd.so` driver does *not* support multicast addresses.

For example, you could use `devn-fd.so` to connect two machines with a null-modem RS-232 serial cable. By using file-descriptor I/O to the serial devices, `devn-fd.so` would implicitly use a serial driver and set up a logical network link.

This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Examples:

Start `io-pkt-v4-hc` using the FD driver:

```
io-pkt-v4-hc -d fd fd=/dev/ser1,mac=0023456789AB,ahdlc
ifconfig en0 10.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

Caveats:

You must specify the fd option when using this driver.

Copyright © 2014, QNX Software Systems Limited

devn-i82544.so

Driver for Intel Gigabit Ethernet LAN adapters and I/O Controller Hubs

Syntax:

```
io-pkt-variant -d /lib/dll/devn-i82544.so
    [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

If you don't specify the full path to `devn-i82544.so`, `io=pkt*` starts *devnp-i82544.so* (p. 432).

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

Detect only devices with the given PCI device ID. The default is automatically detected on supported hardware.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if you specify duplex alone, it's ignored, and both speed and duplex are auto-negotiated.

flowcontrol=0|1

Disable (0) or enable (1) hardware flow control.

irq=*N*

IRQ of the interface. The default is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

mtu=*X*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pauseignore

Ignore pause frames with respect to full duplex flow control.

pausesuppress

Suppress pause frames with respect to full duplex flow control.

pci=*0xXXXX*

Detect devices only at this specific PCI index.

priority=*X*

Priority of the driver's event-handler thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 128.

speed=*10|100|1000*

Media data rate (10Mbit, 100Mbit, or Gigabit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if you specify speed alone, the specified speed is correctly set, but duplex defaults to half (0).

transmit=*num*

Set the number of transmit descriptors. The default is 512.

Copyright © 2014, QNX Software Systems Limited

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Detect only devices with this specific PCI vendor ID.

Description:

The `devn-i82544.so` driver manages the Intel 82540, 82541, 82542, 82543, 82544, 82545, 82546, 82547, 82571, 82572, and 82573 Gigabit Ethernet LAN adapters, and the ICH8 and ICH9 I/O Controller Hubs. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

`ifconfig en`*X*

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the `devn-i82544.so` driver:

```
io-pkt-v4-hc -d /lib/dll/devn-i82544.so
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007)

Copyright © 2014, QNX Software Systems Limited

devn-micrel8841.so

Driver for Micrel 8841 (1 port) or 8842 (2 port) Ethernet controllers

Syntax:

```
io-pkt-variant -d micrel8841 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

The PCI device ID.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are autonegotiated.

lan=*num*

The LAN number of port 0 (8841 or 8842).

lan2=*num*

The LAN number of port 1 (8842).

multicast

Enable the receipt of all multicast packets, all ports. By default, the receipt of multicast packets is disabled.

pci=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

The PCI index of the controller.

port0=*num*

1 means power down port 0 PHY (the default is 0, power on).

port1=*num*

1 means power down port 1 PHY (the default is 0, power on) on the 8842 (2 port).

priority=*N*

The priority of the driver's event thread. The default is 21.

promiscuous

Allow the driver to pass all data packets received, regardless of the address. By default, promiscuous mode is disabled.

receive=*num*

The number of Rx descriptors (and 2 KB npkts) to allocate. The default is 256.

speed=10|100

Set the link speed (specified in Mbits/second: 10 or 100).

switch

Enable switch mode on the 2-port 8842.

This is very dangerous if you loop the network because of the resulting packet storms.

transmit=*num*

The number of Tx descriptors to allocate (the default is 256).

verbose or verbose=*N*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

The PCI vendor ID.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devn-micrel8841.so` driver controls Micrel 8841 (1 port) or 8842 (2 port) Ethernet controllers. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

This driver supports only PCI versions of the Micrel 8841 (1 port) and 8842 (2 port) Ethernet controllers.

This driver supports hardware checksum calculations for both Tx and Rx of IP and TCP packets (UDP isn't supported). To enable hardware checksumming, use *ifconfig* (p. 957) like this, after starting the driver:

```
ifconfig enX ip4csum tcp4csum
```

Examples:

Start `io-pkt-v6-hc` using the `devn-micrel8841.so` driver and Qnet with 1024 transmit descriptors, and 1024 receive descriptors to avoid packet loss due to scheduling latency on slow processors:

```
io-pkt-v6-hc -d micrel8841 transmit=1024,receive=1024 -p qnet
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-ne2000.so

Driver for NE-2000-compatible Ethernet adapters

Syntax:

```
io-pkt-variant -d ne2000 [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

PCI device ID. The default is automatically detected on supported hardware.

ioport=*num*

I/O port of the interface. The *port* parameter must be a hex address (e.g. 0x320). The default is automatically detected on supported hardware (but see caution below).

irq=*num*

IRQ of the interface. The default is automatically detected on supported hardware (but see caution below).

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of the controller. The default is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

Copyright © 2014, QNX Software Systems Limited

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on the supported hardware.

promiscuous

Enable promiscuous mode. The default is off.

tmem=*name*

Name for typed memory.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

vid=*0xXXXX*

The PCI vendor ID of the controller. The default is automatically detected on the supported hardware.

width=*8|16*

I/O access width (8 or 16 bits). The default is automatically detected on supported hardware.

Description:

The `devn-ne2000.so` driver controls NE-2000-compatible Ethernet adapters.

This driver can't always detect the correct irq and ioport options, especially for ISA devices. To be sure, always specify irq and ioport when using this driver.

This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`

Copyright © 2014, QNX Software Systems Limited

- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the NE-2000 driver:

```
io-pkt-v4-hc -d ne2000 ioport=0x320,irq=11
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Caveats:

If you're running a PCMCIA NE2000-compatible adapter, you may need to specify the -w8 command-line option of `devp-pccard`.

Copyright © 2014, QNX Software Systems Limited

devn-pcnet.so

Driver for AMD PCNET (AMD-79c97x) compatible Ethernet adapters

Syntax:

```
io-pkt-variant -d pcnet [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0
BNC

1
UTP

2
AUI

3
FIBER

deviceindex=*0xXXXX*

Only attach to device with this PCI index.

did=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

PCI device ID. The default is automatically detected on supported hardware.

dma=*num*

The DMA channel.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

hpna

Use HPNA phy.

iftype=*num*

Interface type (from <net/if_types.h>). The default is IFT_ETHER.

iorange=*0xXXXXXXXX*

The I/O base address.

irq=*num*

IRQ of the interface.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

MAC address of the controller. The default is automatically detected on supported hardware.

media=*num*

Media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

memrange=*0xXXXXXXXX*

Register base physical memory address.

mmap

Use memory-mapped registers. The default is IO mapped.

mru=*num*

Maximum receive unit. The default is 1514.

Copyright © 2014, QNX Software Systems Limited

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

phy=*num*

Address of the connected PHY device.

priority=*N*

Priority of the driver event-thread. The default is 21.

probe_phy=0|1

Select whether or not to probe the PHY at regular intervals. For the default value of 0, the PHY is polled only at regular intervals when the interface is down or doesn't receive any packets over the polling interval. If you specify 1, the PHY is always probed at regular intervals to see if the duplex and/or speed of the connection has changed.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors/buffers. The default is 64.

speed=*10|100*

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

transmit=*num*

Number of transmit descriptors/buffers. The default is 128.

uptype=*name*

Copyright © 2014, QNX Software Systems Limited

Interface name. The default is “en”.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

PCI vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-pcnet.so` driver controls AMD PCNET (AMD-79c97x) compatible Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

- This driver doesn't support Fiber PCNET cards with the AM79C971KC chip.
- We don't recommend that you use `devn-pcnet.so` on the BCM1250 platform, because it can lose receive interrupts.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the AMD PCNET driver:

```
io-pkt-v4-hc -d pcnet
ifconfig en0 10.1.0.184
```

Files:

Copyright © 2014, QNX Software Systems Limited

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-pegasus.so

Driver for SMC EZ Connect and other USB Ethernet adapters based on the Pegasus chipset

Syntax:

```
io-pkt-variant -d pegasus [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

busnum=*0xXX*

Attach to a specific device on the USB bus number.

devnum=*0xXX*

Attach to a specific USB device address.

duplex=0|1

Half (0) or full (1) duplex mode.

iftype=*num*

Interface type (from <net/if_types.h>). The default is `IFT_ETHER`.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

MAC address of the controller. You must set the MAC address for this controller.

media=*num*

Media type (from <hw/nicinfo.h>). The default is `NIC_MEDIA_802_3`.

Copyright © 2014, QNX Software Systems Limited

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support.

path

Set the device name. The default is `/dev/usbenet0`.

phy=*num*

Address of the connected PHY device.

priority=*N*

Priority of the driver event-thread. The default is 21.

probe_phy=0|1

Select whether or not to probe the PHY at regular intervals. For the default value of 0, the PHY is polled only at regular intervals when the interface is down or doesn't receive any packets over the polling interval. If you specify 1, the PHY is always probed at regular intervals to see if the duplex and/or speed of the connection has changed.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors/buffers. The default is 5.

speed=*10|100*

Force media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

transmit=*num*

Set the number of transmit descriptors/buffers. The default is 10.

uptype=*name*

Copyright © 2014, QNX Software Systems Limited

Interface name. The default is “en”.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

wait=*num*

Wait *num* seconds for the USB stack. The default wait time is 60 seconds.

Description:

The `devn-pegasus.so` driver controls the SMC EZ Connect USB Ethernet adapter and other USB Ethernet adapters based on the Pegasus chipset. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v6-hc` using the Pegasus driver:

```
io-pkt-v6-hc -d pegasus
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

Caveats:

Start the USB stack before using this driver (see `devu-ohci.so` or `devu-uhci.so`)

Copyright © 2014, QNX Software Systems Limited

devn-rtl.so

Driver for Realtek 8139 PCI cards

Syntax:

```
io-pkt-variant -d rtl [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0

BNC

1

UTP

2

AUI

3

FIBER

deviceindex=*0xXXXX*

PCI index of the controller. Only attach to device with this PCI index.

did=*0xXXXX*

The PCI device ID. The default is automatically detected on supported hardware.

dma=*num*

Copyright © 2014, QNX Software Systems Limited

The DMA channel.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

iftype=*num*

Interface type (from <net/if_types.h>). The default is IFT_ETHER.

iorange=*0xXXXXXXXX*

The I/O base address. The *0xXXXXXXXX* parameter must be a hex address (e.g. 0x320). The default is automatically detected on supported hardware.

irq=*num*

IRQ of the interface. The default is automatically detected on supported hardware.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

media=*num*

Media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

memrange=*0xXXXXXXXX*

Register base physical memory address.

mmap

Use memory-mapped registers. The default is IO mapped.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*X*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

phy=*num*

Address of the connected PHY device.

priority=*N*

Priority of the driver-event thread. The default is 21.

probe_phy=0|1

Select whether or not to probe the PHY at regular intervals. For the default value of 0, the PHY is polled only at regular intervals when the interface is down or doesn't receive any packets over the polling interval. If you specify 1, the PHY is always probed at regular intervals to see if the duplex and/or speed of the connection has changed.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 30.

speed=*10|100*

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

uptype=*name*

Interface name. The default is “en”.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

The PCI vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-rtl.so` driver controls Realtek 8139 PCI cards. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the Realtek 8139 PCI card driver:

```
io-pkt-v4-hc -d rtl
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-rtl8150.so

Driver for Realtek 8150 Ethernet dongle

Syntax:

```
io-pkt-variant -d rtl8150 [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

busnum=*0xXX*

The USB bus number.

devnum=*0xXX*

The USB device number.

did=*0xXXXX*

The USB device ID.

duplex=0|1

Half (0) or full (1) duplex mode.

iftype=*num*

The interface type (from <net/if_types.h>). The default is IFT_ETHER.

lan=*num*

The LAN number. The default is 0.

On the SH platform, the lan= option gets overridden. As a workaround, fully specify the vendor ID, device ID, bus number, and device number to the driver when starting (e.g. vid=0x0bda,did=0x8150,busnum=1,devnum=2,lan=2).

Copyright © 2014, QNX Software Systems Limited

mac=*XXXXXXXXXXX*

The MAC address of the adapter. The default is automatically detected on supported hardware.

media=*num*

The media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

mru=*num*

The maximum receive unit. The default is 1514.

mtu=*X*

The maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support. By default, multicast is enabled.

path=*name*

The device name. The default is /dev/usbenet0.

phy=*num*

The address of the connected PHY device.

priority=*N*

The priority of the driver-event thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 5.

speed=10|100

The media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if you specify the speed alone, the speed is correctly set, but duplex defaults to half (0).

transmit=*num*

Set the number of tx descriptors. The default is 10.

Copyright © 2014, QNX Software Systems Limited

uptype=*name*

The interface name. The default is “en”.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

The USB vendor ID of the adapter. The default is automatically detected on supported hardware.

wait=*num*

Wait *num* seconds for the USB stack. The default is 60 seconds.

Description:

The `devn-rtl8150.so` driver controls the Realtek 8150 Ethernet dongle, which is included in SMC2208 USB/Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

- This driver doesn't support multicast and promiscuous modes.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start `io-pkt-v6-hc` using the Realtek driver:

```
io-pkt-v6-hc -drtl8150 verbose &
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-sis9.so

Driver for the SiS900 Ethernet controller and compatibles

Syntax:

```
io-pkt-variant -d sis9 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0
BNC

1
UTP

2
AUI

3
FIBER

deviceindex=*0xXXXX*

PCI index of the controller. Only attach to device with this PCI index.

did=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

The PCI device ID. The default is automatically detected on supported hardware.

dma=*num*

The DMA channel.

duplex=*0|1*

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

iftype=*num*

Interface type (from <net/if_types.h>). The default is IFT_ETHER.

iorange=*0xXXXXXXXX*

The I/O base address. The *0xXXXXXXXX* parameter must be a hex address (e.g. 0x320). The default is automatically detected on supported hardware.

irq=*num*

IRQ of the interface. The default is automatically detected on supported hardware.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

media=*num*

Media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

memrange=*0xXXXXXXXX*

Register base physical memory address.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Copyright © 2014, QNX Software Systems Limited

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

phy=*num*

Address of the connected PHY device.

priority=*N*

Priority of the driver-event thread. The default is 21.

promiscuous

Enable promiscuous mode. The default is off.

receive=*num*

Set the number of receive descriptors. The default is 64.

speed=*10|100*

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

transmit=*num*

Set the number of transmit descriptors. The default is 128.

uptype=*name*

Interface name. The default is “en”.

verbose or verbose=*N*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

vid=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

Vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-sis9.so` driver controls the SiS900 Ethernet controller and compatibles.

You can use *ifconfig* (p. 957) to enable hardware checksums if supported.

Examples:

Start `io-pkt-v6-hc` using the SiS900 driver:

```
io-pkt-v6-hc -d sis9
ifconfig en0 10.1.0.184
```

Files:

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-smc9000.so

Driver for SMC91c9x and SMC91c1xx (SMC9000) Ethernet controllers

Syntax:

```
io-pkt-variant -d smc9000 [,option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2*

Network cable connector type:

0

BNC

1

UTP

2

AUI

The default is UTP.

duplex=*0|1*

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

iftype=*num*

Interface type (from `<net/if_types.h>`). The default is `IFT_ETHER`.

iorange=*0xXXXXXXXX*

Copyright © 2014, QNX Software Systems Limited

The I/O base address. The *OxXXXXXXXX* parameter must be a hex address (e.g. `0x320`). The default is automatically detected on supported hardware (but see caution below).

irq=*num*

IRQ of the interface.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of the controller. You must set the MAC address for this controller.

media=*num*

Media type (from <hw/nicinfo.h>). The default is `NIC_MEDIA_802_3`.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*num*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support.

phy=*num*

Address of the connected PHY device.

pktque=*num*

Set the maximum number of packets that will be queued for transmission in system RAM. The default is 100.

pmmparent=*string*

Override the parent component of the path to register for Power Management.

priority=*N*

Priority of the driver event-thread. The default is 21.

promiscuous

Copyright © 2014, QNX Software Systems Limited

Enable promiscuous mode. The default is off.

speed=*10|100*

Force media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

txcap=*num*

Set the maximum number of packets that will be queued for transmission in on-chip SRAM. On the `91c111` the default is 1, since the amount of on-chip memory is limited to 4 packets. Limiting the number of transmit packets prevents packet transmission from using up all the SRAM, and makes more SRAM available for receive. This prevents packet overruns on receive. For other chips with more on-chip SRAM, the limit is 32 packets.

uptype=*name*

Interface name. The default is “en”.

variant=*name*

Set up the driver for a specific board variant. Currently supported variants are `generic`, `assabet`, `dbpxa250dp,`, `graphicsmaster`, `innovator`, and `pxa250tmdp`.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The default is 0. The output goes to *`slogger`* (p. 1807); invoke *`sloginfo`* (p. 1818) to view it.

width=*8|16|32*

I/O access width (8, 16, or 32 bits). The default is 16.

An incorrect width size can lock up the card or the platform CPU.

If you specify the variant option, the width option may be ignored.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devn-smc9000.so` driver controls SMC 91c9x/91c1xx compatible Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

This driver can't always detect the correct irq and iorange options, especially for ISA devices. To be sure, always specify irq and iorange when using this driver.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the SMC9000 driver:

```
io-pkt-v4-hc -dsmc9000 iorange=0x200,irq=5
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Caveats:

You must specify the mac option when using this driver.

Copyright © 2014, QNX Software Systems Limited

devn-smsc9500.so

Driver for the SMSC9500 USB Ethernet dongle

Syntax:

```
io-pkt-variant -d smsc9500 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

busnum=*0xXX*

The USB bus number.

devnum=*0xXX*

The USB device number.

did=*0xXXXX*

The USB device ID.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are autonegotiated.

iftype=*num*

The interface type (from `<net/if_types.h>`). The default is `IFT_ETHER`.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

Copyright © 2014, QNX Software Systems Limited

The interface address of the controller. The default is automatically detected on supported hardware.

media=*num*

The media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

mru=*num*

The maximum receive unit. The default is 1514.

mtu=*num*

The maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support. By default, multicast is enabled.

path="*name*"

Connect to the specified USB stack. The default is /dev/io-usb/io-usb.

phy=*num*

The address of the connected PHY device.

priority=*N*

The priority of the driver's event thread. The default is 21.

promiscuous

Enable the driver to pass all data packets received, regardless of the address. By default, promiscuous mode is disabled.

receive=*num*

The number of Rx descriptors. The default is 5.

speed=10|100

The media data rate in megabits/second.

transmit=*num*

The number of Tx descriptors. The default is 10.

uptype=*name*

The interface name. The default is en.

Copyright © 2014, QNX Software Systems Limited

verbose or verbose=*N*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

The USB vendor ID.

wait=*num*

The number of seconds to wait for the USB stack. The default is 60 seconds.

Description:

The `devn-smsc9500.so` driver controls the SMSC9500 USB Ethernet dongle. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

`ifconfig en`*X*

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v6-hc` using the SMSC9500 driver:

```
io-pkt-v6-hc -dsmsc9500 verbose &
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-tigon3.so

Driver for TIGON3 (BCM570X) Ethernet controller

Syntax:

```
io-pkt-variant -d tigon3 [option[,option ...]] ...&
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*type*

The network cable connector type:

0

BNC

1

UTP

3

FIBER

The default is automatically detected on supported hardware.

did=*0xXXXX*

The device ID. The default is automatically detected on supported hardware.

duplex=*dup*

Copyright © 2014, QNX Software Systems Limited

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it's ignored, and both speed and duplex are autonegotiated.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

The MAC address of the controller. The default is automatically detected on supported hardware.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

The PCI index of the controller. The default is automatically detected on supported hardware.

priority

The priority of the driver thread. The default is 21.

promiscuous=0|1

If set to 1, enable the driver to pass all data packets received, regardless of address. The default is 0.

single

In a multiple NIC environment, stop after the first detected card. Default is to enable all cards found.

speed=10|100|1000

The media data rate, in megabits per second. The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

The vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-tigon3.so` driver is the Ethernet controller for the Broadcom BCM570X. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

- This driver on the Dell PowerEdge 850 board runs only up to 100 Mbit/s, and not 1000 Mbit/s. Other boards work well at 1000 Mbit/s.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4` using the TIGON3 driver:

```
io-pkt-v4 -d tigon3 verbose
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *`io-pkt*`* (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-tulip.so

Driver for DEC 21x4x (Tulip) compatible Ethernet adapters

Syntax:

```
io-pkt-variant -d tulip [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type:

0
BNC

1
UTP

2
AUI

3
FIBER

The default is automatically detected on supported hardware.

did=*0xXXXX*

PCI device ID. The default is automatically detected on supported hardware.

duplex=*0|1*

Copyright © 2014, QNX Software Systems Limited

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXX*

MAC address of the controller. If no SROM is available, the MAC will default to `00:00:00:00:00:00`.

nomulticast

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

nosrom

Informs the driver that there is no valid connected SROM: the driver will default to using media independent interface (MII), physical media interface (PHY) auto-negotiation. (A valid MAC address must be supplied on the command line.)

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

phyaddr=*num*

Override the MII routines and use the specified PHY address.

pktque=*num*

Limit the number of packets in the queue. The default is 100.

priority=*num*

Priority of the driver thread. The default is 21.

promiscuous

Enable the driver to pass all data packets received, regardless of address. By default, promiscuous mode is disabled.

receive=*num*

Number of receive descriptors/buffers. The default is 64.

Copyright © 2014, QNX Software Systems Limited

single

Use this option if you have multiple Tulip cards in your system and want to configure them differently. The single option tells the driver to stop after configuring the first Tulip card it finds. The order of the search can't be determined because it depends on the PCI server and PCI BIOS used. After the first card has been configured, when the driver is invoked again, it will find and configure the next card in order, and so on until all Tulip cards have been configured. The default is to enable all Tulip cards found.

speed=*10|100*

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

threshold=*N*

Amount of packet data that must be in TX FIFO before transmission is initiated. The range is 0-4. The default is 3. If you observe transmit underruns, set the number to 4.

transmit=*num*

Number of transmit descriptors/buffers. The default is 128.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

vid=*0xXXXX*

The PCI vendor ID of the controller. The default is automatically detected on supported hardware.

Description:

The `devn-tulip.so` driver controls DEC 21x4x (Tulip) compatible Ethernet adapters. This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

When you start a single instance of the Tulip driver on a multiport board (using the pci= option), the board might not function unless this instance is on the first interface. On NICs with multiple interfaces sharing a single SROM, the first NIC is the only one that can read the SROM.

Copyright © 2014, QNX Software Systems Limited

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v6-hc` using the Tulip driver:

```
io-pkt-v6-hc -d tulip
ifconfig en0 10.0.0.184
```

Files:

/dev/io-net

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devn-via-rhine.so

Driver for VIA Rhine Network Interface Cards

Syntax:

```
io-pkt-variant -d via-rhine [option[,option ...]] ...
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

connector=*0|1|2|3*

Network cable connector type.

0

BNC

1

UTP

2

AUI

3

FIBER

deviceindex=*0xXXXX*

Only attach to a device with this PCI index.

did=*0xXXXX*

Copyright © 2014, QNX Software Systems Limited

Device ID. The default is automatically detected on supported hardware.

dma=*num*

DMA channel.

duplex=*0|1*

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

iftype=*num*

Interface type (from <net/if_types.h>). The default is IFT_ETHER.

iorange=*0xXXXXXXXX*

The IO base address.

irq=*num*

IRQ of the interface.

lan=*num*

LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

MAC address of controller. The default is automatically detected on supported hardware.

media=*num*

Media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3

memrange=*0xXXXXXXXX*

Register base physical memory address.

mru=*num*

Maximum receive unit. The default is 1514.

mtu=*X*

Maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Copyright © 2014, QNX Software Systems Limited

Disables the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

PCI index of the controller. The default is automatically detected on supported hardware.

phy=*num*

Address of the connected PHY device.

priority=*N*

Priority of the driver event thread. The default is 21.

promiscuous=*0|1*

If set to 1, enable the driver to pass all data packets received, regardless of address. The default is 0.

receive=*X*

Number of receive descriptors/buffers. The default is 64.

speed=*10|100*

Media data rate (10Mbit or 100Mbit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed will be correctly set, but duplex will default to half (0).

transmit=*num*

Number of transmit descriptors/buffers. The default is 128.

uptype=*name*

Interface name. The default is “en”.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4, the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807), invoke *sloginfo* (p. 1818) to view.

vid=*0xXXXX*

Vendor ID of the controller. The default is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devn-via-rhine.so` driver controls VIA Rhine Network Interface Cards (NICs). This is a legacy `io-net` driver; its interface names are in the form `en`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Examples:

Start `io-pkt-v4-hc` using the VIA Rhine NIC driver:

```
io-pkt-v4-hc -d via-rhine
ifconfig en0 10.1.0.184
```

Files:

`/dev/io-net`

The directory where, by default, drivers and protocol modules add entries. For more information, see the documentation for *io-pkt** (p. 1007).

Copyright © 2014, QNX Software Systems Limited

devnp-asix.so

Driver for the ASIX AX88172, AX88172A, AX88178, AX88772, AX88772A, AX88772B USB Ethernet dongle

Syntax:

```
io-pkt-variant -d asix [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

busnum=*0xXX*

The USB bus number.

devnum=*0xXX*

The USB device number.

did=*0xXXXX*

The USB device ID.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are autonegotiated.

iftype=*num*

The interface type (from <net/if_types.h>). The default is `IFT_ETHER`.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

Copyright © 2014, QNX Software Systems Limited

The interface address of the controller. The default is automatically detected on supported hardware.

media=*num*

The media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

mru=*num*

The maximum receive unit. The default is 1514.

mtu=*num*

The maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support. By default, multicast is enabled.

path="*name*"

Connect to the specified USB stack. The default is /dev/io-usb/io-usb.

phy=*num*

The address of the connected PHY device.

priority=*N*

The priority of the driver's event thread. The default is 21.

promiscuous

Enable the driver to pass all data packets received, regardless of the address. By default, promiscuous mode is disabled.

receive=*num*

The number of Rx descriptors. The default is 5.

rx_flow

Enable receive flow control (default autonegotiated).

speed=10|100|1000

The media data rate in megabits/second.

transmit=*num*

The number of Tx descriptors. The default is 10.

Copyright © 2014, QNX Software Systems Limited

tx_flow

Enable transmit flow control (default autonegotiated).

uptype=*name*

The interface name. The default is `ax`.

verbose or verbose=*N*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

The USB vendor ID.

wait=*num*

The number of seconds to wait for the USB stack. The default is 60 seconds.

Description:

The `devnp-asix.so` driver controls the ASIX AX88172, AX88172A, AX88178, AX88772, AX88772A, and AX88772B USB Ethernet dongle. This is a native `io-net` driver; its interface names are in the form `ax`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig axX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ax0`, use `if_up -p ax0`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start `io-pkt-v6-hc` using the ASIX driver:

```
io-pkt-v6-hc -dasix verbose &
ifconfig ax0 10.1.0.184
```

Copyright © 2014, QNX Software Systems Limited

devnp-bce.so

Driver for Broadcom BCM440x 10/100 Ethernet controllers

Syntax:

```
io-pkt-variant -d bce ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

None

Description:

The `devnp-bce.so` driver manages the Broadcom BCM440x 10/100 Ethernet controllers. This is a ported NetBSD driver; its interface names are in the form `bce`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig bceX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/bce0`, use `if_up -p bce0`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-bce.so` driver:

```
io-pkt-v4 -d bce
ifconfig bce0 10.184
```

Copyright © 2014, QNX Software Systems Limited

devnp-bge.so

Driver for Broadcom 57xx Tigon3 10/100/1000 Mbit Ethernet controllers

Syntax:

```
io-pkt-variant -d bge ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

None

Description:

The `devnp-bge.so` driver manages the Broadcom 57xx Tigon3 10/100/1000 Mbit Ethernet controllers. This is a ported NetBSD driver; its interface names are in the form `bge`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig bgeX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *`waitfor`* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *`if_up -p`* (p. 955) instead; for example, instead of `waitfor /dev/io-net/bge0`, use `if_up -p bge0`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-bge.so` driver:

```
io-pkt-v4 -d bge
ifconfig bge0 10.184
```

Copyright © 2014, QNX Software Systems Limited

devnp-e1000.so

Driver for Intel Gigabit Ethernet controllers

Syntax:

```
io-pkt-variant -d e1000 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

Detect only devices with this specific PCI Device ID. The default is automatically detected on supported hardware.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

You can also use `ifconfig -m` and `ifconfig wm`*X* `media` to set this.

force_link

Force the link speed/duplex. The default is to autonegotiate the advertised speed/duplex.

int_mod=*N*

The interrupt moderation value. The default is 20000 interrupts/sec; a value of zero disables interrupt moderation.

irq=*N*

IRQ of the interface. The default is automatically detected on supported hardware.

Copyright © 2014, QNX Software Systems Limited

mac=*XXXXXXXXXXX*

The MAC address of the controller. The default is automatically detected on supported hardware.

max_read=*N*

The maximum PCIe read request size. *N* must be 128, 256, 512, 1024, 2048, or 4096 bytes.

mtu=*N*

The maximum transmission unit. The default is 1514.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pauseignore

Ignore pause frames with respect to full duplex flow control.

pausesuppress

Suppress pause frames with respect to full duplex flow control.

pci=*0xXXXX*

Detect only devices at this specific PCI index.

priority=*N*

The priority of the driver's event-handler thread (default 21).

receive=*num*

The number of receive descriptors; the default is 512, and the maximum is 4096.

speed=*N*

Set the link speed (specified in Mbits/second). If you specify speed, specify duplex as well; if speed alone is specified, the specified speed is correctly set, but duplex defaults to half (0).

You can also use `ifconfig -m` and `ifconfig wm`*X* `media` to set this.

transmit=*N*

The number of transmit descriptors; the default is 4096, as is the maximum.

tx_reap=*N*

Copyright © 2014, QNX Software Systems Limited

The maximum number of transmit descriptors to reap. The default is 64.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Detect only devices with this specific PCI Vendor ID.

Description:

The `devnp-e1000.so` driver manages all current Intel Gigabit devices. This is a native `io-pkt` driver; its interface names are in the form `wm`*X*, where *X* is an integer.

The `devnp-e1000.so` and *devnp-i82544.so* (p. 432) drivers are similar:

- `devnp-i82544.so` has performance optimizations (TSO and interrupt thresholding options) that may make it a better performing candidate for some hardware
- `devnp-e1000.so` likely supports more hardware

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig wmX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/wm0`, use `if_up -p wm0`.

The SQE (Squelch Test Errors) counter — one of the fields reported by *nicinfo* (p. 1355) — isn't applicable to `devnp-e1000.so`, so this driver uses it in a non-standard way. You can lose packets because:

Copyright © 2014, QNX Software Systems Limited

- you ran out of descriptors (the NIC was able to buffer the packet, but there was no CPU RAM available)

  or:

- the NIC was unable to buffer the packet because it overran its internal Rx FIFO

Other drivers add the two together, but this driver uses the SQE counter for internal Rx FIFO overruns, which generally indicate excessive bus latency, perhaps misconfigured link-level flow control, or even misconfigured Rx FIFO watermarks.

Examples:

Start `io-pkt` using the `devnp-e1000.so` driver and the full TCP/IP stack:

```
io-pkt -d e1000
ifconfig wm0 192.168.0.10
```

Copyright © 2014, QNX Software Systems Limited

devnp-ecm.so

Driver for the CDC ECM USB Ethernet control module

Syntax:

```
io-pkt-variant -d ecm ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

bufsz=*bytes*

The internal buffer size for the AT command (default 2048).

busnum=*num*

Attach to a specific device on the USB bus with the given number.

devnum=*num*

Attach to a specific USB device address.

ign_remove

Ignore the USB removal callback; the user should handle device removal.

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb/io-usb`.

pnp

Keep the driver loaded across device insertion and removal.

When you use the pnp option, the DLL remains loaded and connected to the USB stack. Ethernet interfaces are created as devices are inserted. If you use `ifconfig ecm`*X*` destroy` to remove the last interface, the DLL is unloaded. This means that the driver currently doesn't support being removed and inserted again.

receive=*N*

The number of receive URBs; the default is 64.

Copyright © 2014, QNX Software Systems Limited

transmit=*N*

The number of transmit URBs; the default is 64.

verbose

Be verbose. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

wait=*num*

Wait *num* seconds for the USB stack (default 60 seconds).

Description:

The `devnp-ecm.so` driver manages the CDC ECM USB Ethernet control module. This is a native `io-pkt` driver; its interface names are in the form `ecm`*X*, where *X* is an integer.

Launcher applications that get notifications of device insertions and removals can use the busnum and devnum options. These applications mount the DLL to target specific devices connected to the USB. When the device is removed, it is expected that the launcher will also unmount the interface associated with interface. For `io-pkt*` drivers, you can't use `umount` to unload the DLL, but you can use `ifconfig`'s `destroy` command to unload DLL.

The driver provides an additional interface `serecm`*X* under `/dev` for AT commands. You can send AT commands and receive responses through this interface.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig ecmX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ecm0`, use `if_up -p ecm0`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-ecm.so` driver:

```
io-pkt-v4-hc -d ecm verbose -ptcpip &
ifconfig ecm0 10.184
```

Unload the DLL:

```
ifconfig ecm0 down
ifconfig ecm0 destroy
```


Copyright © 2014, QNX Software Systems Limited

devnp-ecmplus.so

Driver for the CDC ECM/RMNET USB Ethernet control module

Syntax:

```
io-pkt-variant -d ecmplus ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

busnum=*num*

Attach to a specific device on the USB bus with the given number.

classid=*num*

Attach to the specific class ID with the given value in hexadecimal. This is to support drivers that don't use the Communication class. You should also specify the subclassid option.

ctrlbufsz=*num*

Set the buffer size for control transfers (default 1024).

ctrlrxurb=*num*

Set the URB number for control transfers (default 64).

devnum=*num*

Attach to a specific USB device address.

ign_remove

Ignore the USB removal callback; the user should handle device removal.

intf=*num*

Set the minimum scan interface.

Copyright © 2014, QNX Software Systems Limited

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb/io-usb`.

pnp

Keep the driver loaded across device insertion and removal.

When you use the pnp option, the DLL remains loaded and connected to the USB stack. Ethernet interfaces are created as devices are inserted. If you use `ifconfig ecm`*X*` destroy` to remove the last interface, the DLL is unloaded. This means that the driver currently doesn't support being removed and inserted again.

receive=*N*

The number of receive URBs; the default is 64.

subclassid=*num*

Attach to the specific subclass ID with the given value, in hexadecimal. This is to support drivers that don't use the ECM subclass. You should also specify the classid option.

transmit=*N*

The number of transmit URBs; the default is 64.

verbose

Be verbose. The output goes to *`slogger`* (p. 1807); invoke *`sloginfo`* (p. 1818) to view it.

wait=*num*

Wait *num* seconds for the USB stack (default 60 seconds).

Description:

The `devnp-ecmplus.so` driver manages the CDC ECM USB Ethernet control module. This is a native `io-pkt` driver; its interface names are in the form `ecm`*X*, where *X* is an integer.

Launcher applications that get notifications of device insertions and removals can use the busnum and devnum options. These applications mount the DLL to target specific devices connected to the USB. When the device is removed, it is expected that the launcher will also unmount the interface associated with interface. For `io-pkt*` drivers, you can't use `umount` to unload the DLL, but you can use `ifconfig`'s `destroy` command to unload DLL.

Copyright © 2014, QNX Software Systems Limited

The driver provides an additional interface `serecm`*X* under `/dev` for AT commands or vendor-specific commands, such as Qualcomm QMI messages. You can send commands and receive responses through this interface.

For RMNET devices, you should specify the classid and subclassid options.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig ecmX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *`waitfor`* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *`if_up -p`* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ecm0`, use `if_up -p ecm0`.

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-ecmplus.so` driver:

```
io-pkt-v4-hc -d ecmplus verbose -ptcpip &
ifconfig ecm0 10.184
```

Unload the DLL:

```
ifconfig ecm0 down
ifconfig ecm0 destroy
```

Copyright © 2014, QNX Software Systems Limited

devnp-i80579.so

Driver for Intel Tolapai 80579 Gigabit Ethernet controllers

Syntax:

```
io-pkt-variant -d i80579 [option[,option...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

did=*X*

Detect only devices with this specific PCI Device ID (e.g. 0x5041, 0x5045, or 0x5049).

poll

For debugging only. Don't use an interrupt; poll Rx instead.

probe_phy=0|1

Disable (0) or force (1) periodic PHY probing.

receive=*X*

The number of receive descriptors (the default is 256).

transmit=*X*

The number of transmit descriptors (the default is 256).

verbose=*N*

Set the verbosity level. The default is zero; a larger value for *N* yields more output. The output is sent to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devnp-i80579.so` driver manages the Intel Tolapai 80579 Gigabit Ethernet controller. This is a native `io-pkt` driver; its interface names are in the form `gbe`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig gbeX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *`waitfor`* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *`if_up -p`* (p. 955) instead; for example, instead of `waitfor /dev/io-net/gbe0`, use `if_up -p gbe0`.

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-i80579.so` driver:

```
io-pkt-v4 -d i80579
ifconfig gbe0 10.1
ping -n 10.2
```

Copyright © 2014, QNX Software Systems Limited

devnp-i82544.so

Driver for Intel 825 Gigabit Ethernet LAN adapters*

Syntax:

```
io-pkt-variant -d i82544 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

Detect only devices with this specific PCI Device ID. The default is automatically detected on supported hardware.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

You can also use `ifconfig -m` and `ifconfig wm`*X* `media` to set this.

irq=*N*

IRQ of the interface. The default is automatically detected on supported hardware.

irq_thresh=*X*

The maximum number of interrupts per second to be generated. The default is 9000. This reduces CPU consumption by limiting how often interrupts occur, which is especially helpful on slower processors.

Copyright © 2014, QNX Software Systems Limited

mac=*XXXXXXXXXXX*

The MAC address of the controller. The default is automatically detected on supported hardware.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pause_rx_disable

Disregard received pause (flow control) frames.

pause_rx_enable

Always act on received pause (flow control) frames.

pause_tx_disable

Never transmit pause (flow control) frames.

pause_tx_enable

Always transmit pause (flow control) frames.

pci=*0xXXXX*

Detect only devices at this specific PCI index.

receive=*num*

The number of receive descriptors; the default is 512.

rx_abs=*X*

The receive interrupt absolute delay time multiplier. The default is 92, and the maximum is 200.

rx_delay=*X*

The receive interrupt delay time multiplier (×1.024 ns). The default is 23, and the maximum is 50.

speed=10|100|1000

The media data rate (10Mbit, 100Mbit, or Gigabit operation). The default is automatically detected on supported hardware. If you specify speed, specify duplex as well; if speed alone is specified, the specified speed is correctly set, but duplex defaults to half (0).

You can also use `ifconfig -m` and `ifconfig wm`*X* `media` to set this.

transmit=*N*

Copyright © 2014, QNX Software Systems Limited

The number of transmit descriptors; the default is 4096.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Detect only devices with this specific PCI Vendor ID.

Description:

The `devnp-i82544.so` driver manages the Intel 82540, 82541, 82544, 82545, 82546, 82547, 82571, and 82572 Gigabit Ethernet LAN adapters. This is a native `io-pkt` driver; its interface names are in the form `wm`*X*, where *X* is an integer.

The *devnp-e1000.so* (p. 420) and `devnp-i82544.so` drivers are similar:

- `devnp-i82544.so` has performance optimizations (TSO and interrupt thresholding options) that may make it a better performing candidate for some hardware
- `devnp-e1000.so` likely supports more hardware

Transmit Segmentation Offload (TSO) allows the stack to send very large TCP buffers down to the driver, and the driver takes care of carving them up into individual IP packets of the right size. This can greatly reduce the CPU usage for transmitting large amounts of data.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig wmX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/wm0`, use `if_up -p wm0`.

Copyright © 2014, QNX Software Systems Limited

The SQE (Squelch Test Errors) counter — one of the fields reported by *nicinfo* (p. 1355) — isn't applicable to devnp-i82544.so, so this driver uses it in a non-standard way. You can lose packets because:

- you ran out of descriptors (the NIC was able to buffer the packet, but there was no CPU RAM available)

  or:

- the NIC was unable to buffer the packet because it overran its internal Rx FIFO

Other drivers add the two together, but this driver uses the SQE counter for internal Rx FIFO overruns, which generally indicate excessive bus latency, perhaps misconfigured link-level flow control, or even misconfigured Rx FIFO watermarks.

Examples:

Start io-pkt using the devnp-i82544.so driver and the full TCP/IP stack:

```
io-pkt -d i82544 -p tcpip
ifconfig wm0 10.1.0.184
```

Copyright © 2014, QNX Software Systems Limited

devnp-ixgbe.so

Driver for Intel 10 Gigabit Ethernet controllers

Syntax:

```
io-pkt-variant -d ixgbe [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

Detect only devices with this specific PCI Device ID. The default is automatically detected on supported hardware.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are auto-negotiated.

You can also use `ifconfig -m` and `ifconfig ix`*X* `media` to set this.

force_link

Force the link speed/duplex. The default is to autonegotiate the advertised speed/duplex.

irq=*N*

The IRQ of the interface. The default is automatically detected on supported hardware.

mac=*XXXXXXXXXXXX*

Copyright © 2014, QNX Software Systems Limited

The MAC address of the controller. The default is automatically detected on supported hardware.

mtu=*N*

The maximum transmission unit. The default is 1514.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pauseignore

Ignore pause frames with respect to full duplex flow control.

pausesuppress

Suppress pause frames with respect to full duplex flow control.

pci=*0xXXXX*

Detect only devices at this specific PCI index.

priority=*N*

The priority of the driver's event-handler thread (default 21).

promiscuous

Enable the reception of all packets.

receive=*num*

The number of receive descriptors; the default is 512, and the maximum is 4096.

speed=*N*

Set the link speed (specified in Mbits/second).

transmit=*N*

The number of transmit descriptors; the default is 4096, as is the maximum.

tx_reap=*N*

The maximum number of transmit descriptors to reap. The default is 64.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

Copyright © 2014, QNX Software Systems Limited

vid=*0xXXXX*

Detect only devices with this specific PCI Vendor ID.

Description:

The `devnp-ixgbe.so` driver manages Intel 10 Gigabit Ethernet controllers. This is a native `io-pkt` driver; its interface names are in the form `ix`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

`ifconfig ix`*X*

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *`ifconfig`* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *`waitfor`* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *`if_up -p`* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ix0`, use `if_up -p ix0`.

Examples:

Start `io-pkt` using the `devnp-ixgbe.so` driver:

```
io-pkt-v4 -d ixgbe
ifconfig ix0 192.168.0.10
```


Copyright © 2014, QNX Software Systems Limited

devnp-msk.so

Driver for Marvell Yukon-2 based Gigabit Ethernet adapters

Syntax:

```
io-pkt-variant -d msk ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Targets:

x86

Options:

None

Description:

The `devnp-msk.so` driver supports the Marvell Yukon-2 based Gigabit Ethernet adapters, including the following:

- Marvell Yukon 88E8035, copper adapter
- Marvell Yukon 88E8036, copper adapter
- Marvell Yukon 88E8038, copper adapter
- Marvell Yukon 88E8050, copper adapter
- Marvell Yukon 88E8052, copper adapter
- Marvell Yukon 88E8053, copper adapter
- Marvell Yukon 88E8055, copper adapter
- SK-9E21 1000Base-T single port, copper adapter
- SK-9E22 1000Base-T dual port, copper adapter
- SK-9E81 1000Base-SX single port, multimode fiber adapter
- SK-9E82 1000Base-SX dual port, multimode fiber adapter
- SK-9E91 1000Base-LX single port, single mode fiber adapter
- SK-9E92 1000Base-LX dual port, single mode fiber adapter
- SK-9S21 1000Base-T single port, copper adapter
- SK-9S22 1000Base-T dual port, copper adapter
- SK-9S81 1000Base-SX single port, multimode fiber adapter
- SK-9S82 1000Base-SX dual port, multimode fiber adapter

Copyright © 2014, QNX Software Systems Limited

- SK-9S91 1000Base-LX single port, single mode fiber adapter
- SK-9S92 1000Base-LX dual port, single mode fiber adapter
- SK-9E21D 1000Base-T single port, copper adapter

This is a ported NetBSD driver; its interface names are in the form `msk`*X*, where *X* is an integer.

Support for jumbo frames is provided via the interface MTU setting. Selecting an MTU larger than 1500 bytes with the *ifconfig* (p. 957) utility configures the adapter to receive and transmit jumbo frames. Using jumbo frames can greatly improve performance for certain tasks, such as file transfers and data streaming.

Hardware TCP/IP checksum offloading for IPv4 is supported.

The following media types and options (as given to `ifconfig`) are supported:

media autoselect

Enable the autoselection of the media type and options. You can manually override the autoselected mode.

media 1000baseSX mediaopt full-duplex

Set 1000Mbps (Gigabit Ethernet) operation on fiber and force full-duplex mode.

media 1000baseSX mediaopt half-duplex

Set 1000Mbps (Gigabit Ethernet) operation on fiber and force half-duplex mode.

media 1000baseT mediaopt full-duplex

Set 1000Mbps (Gigabit Ethernet) operation and force full-duplex mode.

For more information on configuring this device, see *ifconfig* (p. 957). To view a list of media types and options supported by the card, try `ifconfig -m` *device*. For example, `ifconfig -m msk0`.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/msk0`, use `if_up -p msk0`.

Copyright © 2014, QNX Software Systems Limited

devnp-ncm.so

Driver for the USB CDC NCM network control module

Syntax:

```
io-pkt-variant -d ncm ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

bufsz=*bytes*

The internal buffer size for the AT command (default 2048).

busnum=*num*

Attach to a specific device on the USB bus with the given number.

devnum=*num*

Attach to a specific USB device address.

ext_name

Add the bus number and device number to the network interface name.

ign_remove

Ignore the USB removal callback; the user should handle device removal.

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb/io-usb`.

pnp

Keep the driver loaded across device insertion and removal.

When you use the pnp option, the DLL remains loaded and connected to the USB stack. Ethernet interfaces are created as devices are inserted. If you use `ifconfig ncm`*X*` destroy` to remove the last interface, the DLL is unloaded. This means the driver currently doesn't support being removed and inserted again.

Copyright © 2014, QNX Software Systems Limited

receive=*N*

The number of receive URBs; the default is 64.

transmit=*N*

The number of transmit URBs; the default is 64.

verbose

Be verbose. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

wait=*num*

Wait *num* seconds for the USB stack (default 60 seconds).

Description:

The `devnp-ncm.so` driver manages the USB CDC NCM network control module. This is a native `io-pkt` driver; its interface names are in the form `ncm`*X*, where *X* is an integer.

Launcher applications that get notifications of device insertions and removals can use the busnum and devnum options. These applications mount the DLL to target specific devices connected to the USB. When the device is removed, it is expected that the launcher will also unmount the interface associated with interface. For `io-pkt*` drivers, you can't use `umount` to unload the DLL, but you can use `ifconfig`'s `destroy` command to unload DLL.

The driver provides an additional interface `serncm`*X* under `/dev` for AT commands. You can send AT command and receive response through this interface.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig ncmX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Copyright © 2014, QNX Software Systems Limited

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ncm0`, use `if_up -p ncm0`.

Examples:

Start the v4 TCP/IP variant of `io-pkt` using the `devnp-ncm.so` driver:

```
io-pkt-v4-hc -d ncm verbose -ptcpip &
ifconfig ncm0 10.184
```

Unload the DLL:

```
ifconfig ncm0 down
ifconfig ncm0 destroy
```

Copyright © 2014, QNX Software Systems Limited

devnp-rtl8169.so

Driver for Realtek 8169 Gigabit Ethernet controllers

Syntax:

```
io-pkt-variant -d rtl8169 [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

The PCI device ID.

duplex=0|1

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, specify speed as well; if duplex alone is specified, it is ignored and both speed and duplex are autonegotiated.

iftype=*num*

The interface type (from `<net/if_types.h>`). The default is `IFT_ETHER`.

iorange=*0xXXXXXXXX*

The I/O base address.

irq=*num*

The IRQ of the interface.

lan=*num*

The LAN number. The default is 0.

mac=*XXXXXXXXXXXX*

Copyright © 2014, QNX Software Systems Limited

The interface address of the controller. The default is automatically detected on supported hardware.

media=*num*

The media type (from <hw/nicinfo.h>). The default is NIC_MEDIA_802_3.

mru=*num*

The maximum receive unit. The default is 1514.

mtu=*num*

The maximum transmission unit. The default (1514) is automatically detected on supported hardware.

nomulticast

Disable multicast support. By default, multicast is enabled.

pci=*0xXXXX*

The PCI index of the controller.

phy=*num*

The address of the connected PHY device.

priority=*N*

The priority of the driver's event thread. The default is 21.

receive=*num*

The number of Rx buffers to internally cache. The default is 5.

speed=10|100|1000

The media data rate in megabits/second.

transmit=*num*

The number of Tx buffers to internally cache. The default is 10.

uptype=*name*

The interface name. The default is en.

verbose or verbose=*N*

Copyright © 2014, QNX Software Systems Limited

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The default is 0. The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

The PCI vendor ID.

Description:

The `devnp-rtl8169.so` driver controls Realtek 8169 Gigabit Ethernet controllers. This is a native `io-pkt` driver; its interface names are in the form `rt`*X*, where *X* is an integer.

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig enX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/rt0`, use `if_up -p rt0`.

Examples:

Start `io-pkt-v4` using the Realtek driver:

```
io-pkt-v4 -d rtl8169
ifconfig rt0 10.184
```


Copyright © 2014, QNX Software Systems Limited

devnp-shim.so

"Shim" driver for backward compatibility with `io-net`

Syntax:

`io-pkt-`*`variant`* `-d shim` *`options`* `"`*`io-net_drvr [drvr_opt,...]`*`"`

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

shimrxcopy=0|1

Specify how packets are copied:

- 1 — copy each received packet in the shim layer as it's passed from the driver to `io-pkt`. You can safely remove the driver and shim via the `ifconfig en`*`X`* `destroy` command or by physically removing the device (if supported by the driver). This is the default.
- 0 — pass each received packet from the driver through the shim directly to `io-pkt`. This may improve performance, but driver and shim removal is blocked.

Description:

The `devnp-shim.so` shared object is a "shim" driver that lets *`io-pkt`* (p. 1007) support `devn-*` drivers that were written for `io-net`.

Explicitly loading the shim driver is usually unnecessary; `io-pkt` loads the shim automatically. For example, if you type:

```
io-pkt -d some_driver
```

then `io-pkt` searches for `devnp-some_driver.so` and loads it as a native driver if found. If not found, `io-pkt` tries to load `devn-some_driver.so` via the shim.

If you type:

```
io-pkt -d /lib/dll/devn-some_driver.so
```

then `io-pkt` notices that the driver is an `io-net` one and loads it via the shim.

Copyright © 2014, QNX Software Systems Limited

- Shim drivers name their interface entries en*X*, but native drivers use a naming scheme that depends on the chipset.
- You can tell if the shim has been loaded by using `pidin me`.

Examples:

```
io-pkt -d shim devn-epic.so
io-pkt -d shim "/lib/dll/devn-epic.so transmit=1024,receive=1024"
```


Copyright © 2014, QNX Software Systems Limited

devnp-speedo.so

Driver for Intel 82557, 82558, and 82559 Fast Ethernet LAN adapters

Syntax:

```
io-pkt-variant -d speedo [[index:option[,[index:option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

did=*0xXXXX*

Device ID. Only attach to device with this PCI index. The default is automatically detected on supported hardware.

duplex=*0|1*

Half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you specify duplex, you must also specify speed.

irq=*num*

The IRQ of the interface. The default is automatically detected on supported hardware.

kermask=0|1

Specify the masking:

- 1 — use the kernel interrupt-masking methodology.
- 0 — manually mask the NIC in the interrupt handler.

mac=*XXXXXXXXXXX*

The MAC address of the controller. The default is automatically detected on supported hardware.

mmap

Copyright © 2014, QNX Software Systems Limited

Use memory-mapped registers. The default is I/O mapped.

The mmap option is supported on all targets except x86.

nomulticast

Disable the driver from sending or receiving multicast packets. By default, multicast is enabled.

pci=*0xXXXX*

The PCI index of the controller. The default is automatically detected on supported hardware.

phy=*num*

The address of the connected PHY device.

receive=*num*

The number of receive descriptors; the default is 256.

speed=10|100

The media data rate (10 Mbit or 100 Mbit operation). The default (0) is automatically detected on supported hardware. If you specify speed, you must also specify duplex.

transmit=*num*

The number of transmit descriptors; the default is 1024.

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *slogger* (p. 1807); invoke *sloginfo* (p. 1818) to view it.

vid=*0xXXXX*

Attach only to devices with this PCI vendor ID. The default is `0x8086`.

Description:

The `devnp-speedo.so` driver manages the Intel 82557, 82558, and 82559 Fast Ethernet LAN adapters. This is a native `io-pkt` driver; its interface names are in the form `fxp`*X*, where *X* is an integer.

Copyright © 2014, QNX Software Systems Limited

Some devices support hardware checksums, although some might do so in only one direction; to determine if your device does, type:

```
ifconfig fxpX
```

and look for the following in the list of supported options:

- `ip4csum`, `ip4csum-rx`, `ip4csum-tx`
- `tcp4csum`, `tcp4csum-rx`, `tcp4csum-tx`
- `tcp6csum`, `tcp6csum-rx`, `tcp6csum-tx`
- `udp4csum`, `udp4csum-rx`, `udp4csum-tx`
- `udp6csum`, `udp6csum-rx`, `udp6csum-tx`

You can then use *ifconfig* (p. 957) to enable or disable whichever of these options your device supports.

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/fxp0`, use `if_up -p fxp0`.

Examples:

Start `io-pkt` using the `devnp-speedo.so` driver and the full TCP/IP stack:

```
io-pkt -d speedo -p tcpip
ifconfig fxp0 10.1.0.184
```

For the second instance of the device in the system, start `io-pkt` using the `devnp-speedo.so` driver and the full TCP/IP stack. Use increased verbosity and override the default #MAC address:

```
io-pkt -d speedo verbose,idx1:mac=00:03:02:01:00:00 -p tcpip
ifconfig fxp0 10.1.0.184
```

Copyright © 2014, QNX Software Systems Limited

devnp-usbdnet.so

Class Driver for USBDNET (USB Device Network Driver)

Syntax:

```
io-pkt-variant -d usbdnet [option[,option ...]] ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

Use commas, not spaces, to separate the options.

iface_num=*number*

An interface delimiter that marks the start of a new set of interface parameters when you're configuring multiple interfaces.

mac=*XXXXXXXXXXXX*

The MAC address on the QNX Neutrino side; a 12-character hexadecimal string (e.g., `0123456789cd`).

mtu=*num*

The maximum transmission unit (default 1500).

nodescmod

Don't modify the USB descriptors; let a USB manager do that.

npkt=*number*

The maximum number of Ethernet frames to bundle (for RNDIS and NCM only).

path=*name*

Connect to the specified USB stack. The default is `/dev/io-usb-dcd/io-usb`.

protocol=*str*

Copyright © 2014, QNX Software Systems Limited

The protocol to use; one of the following:

- ecm — Ethernet Control Model
- rndis — Remote Network Driver Interface Specification
- ncm — Network Control Model

The default is ecm. The interface name matches the protocol selected.

receive=*num*

The number of receive URBs to use; the default is 32.

transmit=*num*

The number of transmit URBs to use; the default is 32.

usbdnet_mac=*XXXXXXXXXXXX*

The MAC address on the host side; a 12-character hexadecimal string (e.g., `0123456789ab`).

verbose or verbose=*num*

Be verbose. Specify *num* for more verbosity (*num* can be 1-4; the higher the number, the more detailed the output). The output goes to *`slogger`* (p. 1807); invoke *`sloginfo`* (p. 1818) to view it.

wait=*num*

Wait *num* seconds for the USB stack (default 60 seconds).

Description:

The `devnp-usbdnet.so` driver is the class driver for USBDNET (USB Device Network Driver). This is a native `io-pkt` driver; its interface names are in the form `ecm`*X*, `ncm`*X*, or `rndis`*X*, depending on the protocol, where *X* is an integer.

- You can set the MTU only on the command line (e.g., mtu=8100), and not using *`ifconfig`* (p. 957). This is because some protocols (such as ECM) require that you advertise the MTU in the device descriptor. Changing the MTU using `ifconfig` would require the driver to force the host to reenumerate the device.
- You can destroy the interface only when the USB cable is disconnected; otherwise the `ifconfig if` *ifaceX* `destroy` command will fail.

Copyright © 2014, QNX Software Systems Limited

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *`waitfor`* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *`if_up -p`* (p. 955) instead; for example, instead of `waitfor /dev/io-net/ecm0`, use `if_up -p ecm0`.

Examples:

Start `io-pkt` using the `devnp-usbdnet.so` driver and the default ECM protocol:

```
io-pkt-v4-hc -d usbdnet -ptcpip &
ifconfig ecm0 192.168.1.1
```


Copyright © 2014, QNX Software Systems Limited

devp-pccard

PCMCIA/CardBus (PC Card) server

You must be `root` to start this driver.

Syntax:

```
devp-pccard [Card Services options] [ss Socket Services options]...
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

Card Services Options

The card services options include the following:

-a

The ioport address to be assigned to a PCMCIA card. Use a colon (:) to separate functions on multi-function cards. E.g. `-a 0x300:0x320,0x340` will assign ioport `0x300` to function 1 in socket 0 and ioport `0x320` to function 2 in socket 0; ioport `0x340` will be assigned to function 1 in socket 1.

-i *irq*

The IRQ to be used for status interrupts. The default is no interrupts — the adapter is polled every second for status changes (recommended).

-l

("el") Override PCMCIA IRQ for socket(s). E.g. `-l5` will assign IRQ 5 to the card in socket 0; `-l5,7` will assign IRQ5 to the card in socket 0 and IRQ 7 to the card in socket 1.

-m

Copyright © 2014, QNX Software Systems Limited

Memory window address for reading CIS. (Default is 0xd4000)

-v

Verbose output for debugging purposes.

-w

Force the width for a PCMCIA socket (8 or 16 bits). E.g. `-w8` will force an 8-bit width for socket 0; `-w16,8` will force a 16-bit width for socket 0 and an 8-bit width for socket 1. This is needed by some Ethernet adapters that report themselves as 16-bit, but work only in 8-bit mode.

-x *index*

Select the PCMCIA configuration index to use. Some PCMCIA cards have multiple configuration indexes. This option can be used to select one of them.

Socket Services options

The socket services options include the following:

-D *Device ID*

Specify the PCI device ID to which `devp-pccard` must attach. This option *must* be used in conjunction with the -V option.

-I *index*

Specify the PCI index to which `devp-pccard` must attach.

-m

Map ISA interrupts to PCI bus.

-n

Set hardware interrupt routing on PCI bus. Note: This may not work with some BIOSes.

-p

Set the IRQ mode (0 - 3) as follows:

Mode	Sets
0	Parallel PCI interrupts only

Copyright © 2014, QNX Software Systems Limited

Mode	Sets
1	Parallel IRQ and parallel PCI interrupts
2	IRQ serialized interrupts and parallel PCI interrupts
3	IRQ and PCI serialized interrupts

-r

Value to set in multi-function routing register (chip specific).

-V ***Vendor ID***

Specify the PCI vendor ID to which `devp-pccard` must attach. This option *must* be used in conjunction with the -D option.

-v

Verbosity for socket services.

Description:

The `devp-pccard` server provides support under QNX Neutrino for PCMCIA and CardBus host adapter chips. The host adapters chips currently supported are (PCMCIA) Intel 82365, Cirrus CL-PD67*xx*, Vadem VG-46*x*, (CardBus) TI-11*xx*, TI-12*xx*, and TI-14*xx*, Ricoh R5C47*x*, O2 Micro OZ68*xx*, and Toshiba Topic97. Other CardBus adapters work only in legacy (PCMCIA) mode.

The server manages host resources (memory windows, I/O ports and IRQs) and assigns resources to PCMCIA cards as they're inserted. CardBus resources are managed by the *pci-bios* (p. 1455) server, which interfaces to the pccard server. The `devp-pccard` server also supports dual-function PC Cards and assigns separate resources to each function. The only common resource assigned to dual-function PC Cards is the IRQ.

Utilities are provided to start and stop processes (as cards are inserted and removed), display server status and display card CIS (Card Information Structure) data.

The executables involved in PC Card support are:

`devp-pccard`

The server for PCMCIA and CardBus adapters.

pccard-launch **(p. 1451)**

Copyright © 2014, QNX Software Systems Limited

A manager that starts and stops processes as cards are inserted and removed.

pin **(p. 1541)**

A utility that displays PC Card information (CIS, status, and so on).

Resources and Server Configuration Files

The server manages separate resource pools for memory windows, IRQs and ports. When a card is installed, the server attempts to satisfy the card's memory window, IRQ and port requirements by allocating resources from the various pools. PCMCIA resources must be in the ISA range of devices, while CardBus resources must be in the PCI range. PC Card resource pools are created as described below.

CardBus

CardBus resources are assigned by the *pci-bios* (p. 1455) server, as all CardBus devices are considered to be PCI devices. Some manufacturers' PCI BIOSes allow separate IRQs to be assigned to each socket on a CardBus adapter, while others allow only a single IRQ for both sockets. When a CardBus PC Card is inserted in a socket, `devp-pccard` requests the `pci-bios` server to rescan its bus and allocate resources to the card.

Examples:

Start devp-pccard and force the ioports to be used in each socket:

```
devp-pccard -a 0x300,0x340
```

Map IRQ interrupts to PCI and use IRQ 10:

```
devp-pccard -l10 ss -m
```


Copyright © 2014, QNX Software Systems Limited

devu-ehci.so

Driver for Enhanced Host Controller Interface (EHCI) USB controllers

You must be `root` to start this driver.

Syntax:

```
io-usb -d ehci [option[,option...]] &
```

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

en_sched

Always enable the scheduler. By default, the scheduler is enabled when there are devices connected.

frame_list_size=*num*

Set the size of the frame list: 1024 (the default), 512, or 256.

int_thresh=*num*

Interrupt microframe threshold. Valid values are 1, 2, 4, 8, 16, 32, and 64. The default value is 8, which is a 1 ms interrupt rate. The chip will interrupt after a 1 ms period (USB frame) if there are completed transactions. Setting the value to 1 lets the chip interrupt after 125 us (USB microframe) if there are completed transactions.

ioport=*addr*

Register the base address. The default is to scan the PCI bus.

irq=*num*

The interrupt request number.

memory=*name*

Copyright © 2014, QNX Software Systems Limited

Use the specified typed memory for DMA descriptors (endpoint descriptor, transfer descriptors, and so on).

nosmm

Don't disable system management. The default is to disable it.

num_ed=*num*

The number of endpoint descriptors to preallocate.

num_itd=*num*

The number of Isoch transfer descriptors to preallocate; the default is to use the global transfer descriptors pool as specified by the num_td option.

You must specify the num_itd option after the ioport and irq options.

num_td=*num*

The number of transfer descriptors to preallocate.

pindex=*num*

Instance of controller on PCI bus to apply argument.

ports=*port*:*port*...

Set the enumeration order of each root port. Use colons to separate the port numbers.

soft_retries=*num*

The number of software retries for transaction errors. The default is 6.

verbose=*num*

Set verbosity level.

Description:

The `devu-ehci.so` server provides support for computers that have Enhanced Host Controller Interface (EHCI) USB controllers. This server creates the `/dev/io-usb/devu-ehci.so` device.

Examples:

Scan PCI bus for all available controllers.

```
io-usb -dehci
```


Copyright © 2014, QNX Software Systems Limited

If you specify arguments you'll need to include *pindex* arguments for all controllers you wish to use. The value for the *pindex* argument should start from 0. Attach to the 1st and 3rd controller instance and set the *verbose* argument on the 1st, as follows:

```
io-usb -dehci pindex=0,verbose=4,pindex=2 &
```

Copyright © 2014, QNX Software Systems Limited

devu-kbd

Class driver for USB keyboards (BOOT mode HID)

You must be logged in as `root`, and start a USB stack (see *io-usb* (p. 1015)) before you start this driver.

Syntax:

```
devu-kbd [options*] &
```

Runs on:

QNX Neutrino

Options:

-n *name*

The device name to use. The default is `/dev/usbkbd0`.

-s *stack*

The name of the stack to attach to (default: `/dev/io-usb/io-usb`).

-v

Be verbose.

-w *secs*

Wait *sec* seconds for the USB stack (default: 60 seconds).

Description:

The `devu-kbd` class driver manages USB keyboards.

See *devu-mouse* (p. 463) for information on USB mice.

Examples:

```
devu-kbd &
```


Copyright © 2014, QNX Software Systems Limited

devu-mouse

Class driver for USB mice (BOOT mode HID)

You must be logged in as `root`, and start a USB stack (see *io-usb* (p. 1015)) before you start this driver.

Syntax:

```
devu-mouse [options*] &
```

Runs on:

QNX Neutrino

Options:

-n *name*

The device name to use. The default is `/dev/usbmouse0`.

-s *stack*

The name of the stack to attach to (default: `/dev/io-usb/io-usb`).

-v

Be verbose.

-w *secs*

Wait *sec* seconds for the USB stack (default: 60 seconds).

Description:

The `devu-mouse` class driver manages USB mice.

See *devu-kbd* (p. 462) for information on USB keyboards.

Examples:

```
devu-mouse &
```

Copyright © 2014, QNX Software Systems Limited

devu-ohci.so

Driver for Open Host Controller Interface (OHCI) USB controllers

You must be `root` to start this driver.

Syntax:

```
io-usb -d ohci [option[,option...]] &
```

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

ioport=*addr*

Register the base address of the USB controller. The default is to scan the PCI bus.

irq=*num*

The interrupt request number.

isoptd=*num*

Restrict the number of isoch frames each TD can transfer (default 8).

memory=*name*

Use the specified typed memory for DMA descriptors (endpoint descriptor, transfer descriptors, and so on).

nosmm

Don't disable system management. The default is to disable it.

num_ed=*num*

The number of endpoint descriptors to preallocate. The number specified is added to the minimum number of endpoint descriptors needed to set up the USB chip.

Copyright © 2014, QNX Software Systems Limited

num_td=*num*

The number of transfer descriptors to preallocate. The number specified is added to the minimum number of transfer descriptors needed to set up the USB chip.

pindex=*num*

Instance of controller on PCI bus to apply argument.

verbose=*num*

Set verbosity level.

Description:

The `devu-ohci.so` server provides support for computers that have Open Host Controller Interface (OHCI) USB controllers. This server creates the `/dev/io-usb/devu-ohci.so` device.

Examples:

Scan PCI bus for all available controllers.

```
io-usb -dohci
```

If you specify arguments you'll need to include *pindex* arguments for all controllers you wish to use. The value for the *pindex* argument should start from 0. Attach to the 1st and 3rd controller instance and set the *verbose* argument on the 1st, as follows:

```
io-usb -dohci pindex=0,verbose=4,pindex=2 &
```

Copyright © 2014, QNX Software Systems Limited

devu-prn

Class driver for USB printers

You must be logged in as `root` and start a USB stack (see *io-usb* (p. 1015)) before you start this driver.

Syntax:

```
devu-prn [-m size] [-n name] [-s stack] [-v] [-w secs] &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-m *size*

Specify the size, in bytes, of the maximum output buffer used to write data to the USB stack. The default is 4096 bytes. Larger values may be limited by the I/O buffer limits of the USB chipset you're using.

-n *name*

Set the device name. The default is `/dev/usbpar0`.

-s *stack*

Set the USB stack name. The default is `/dev/io-usb/io-usb`.

-v

Be verbose.

-w *secs*

Wait up to the indicated number of seconds for `/dev/`*stack* to appear. This is useful at boot time for slow-resetting devices. The default is 60 seconds.

Description:

The `devu-prn` class driver manages USB printers.

Copyright © 2014, QNX Software Systems Limited

This manager terminates only upon receipt of a signal or on encountering a problem during startup (e.g. it can't locate the USB stack).

See *devu-ehci.so* (p. 459), *devu-ohci.so* (p. 464), *devu-uhci.so* (p. 468), or *devu-xhci.so* (p. 474) for information on USB stacks. For more information on setting up USB printers, see Connecting Hardware and Printing in the QNX Neutrino *User's Guide*.

Examples:

Start `devu-prn` with a maximum output buffer of 8192 bytes:

```
devu-prn -m 8192 &
```

Copyright © 2014, QNX Software Systems Limited

devu-uhci.so

Driver for Universal Host Controller Interface (UHCI) USB controllers

You must be `root` to start this manager.

Syntax:

```
io-usb -d uhci [option[,option...]] ... &
```

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

ioport=*addr*

Register the base address of the USB controller. The default is to scan the PCI bus.

irq=*num*

The interrupt request number.

nosmm

Don't disable system management. The default is to disable it.

num_ed=*num*

The number of endpoint descriptors to preallocate.

num_td=*num*

The number of transfer descriptors to preallocate.

pindex=*num*

Instance of controller on PCI bus to apply argument.

verbose=*num*

Set verbosity.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devu-uhci.so` server provides support for computers that have Universal Host Controller Interface (UHCI) USB controllers. This server creates the `/dev/io-usb/devu-uhci.so` device.

Examples:

Scan PCI bus for all available controllers.

```
io-usb -duhci
```

If you specify arguments, you must include a *pindex* argument for all controllers you wish to use. The value of the *pindex* argument starts from 0.

Attach to the 1st and 3rd controller instance and set the *verbose* argument on the 1st controller instance as follows:

```
io-usb -duhci pindex=0,verbose=4,pindex=2 &
```

Copyright © 2014, QNX Software Systems Limited

devu-umass_client-block

Function driver for USB mass storage devices

You must be logged in as `root`, and start the USB DCD stack (see *io-usb-dcd* (p. 1018)) before you start this driver.

Syntax:

```
devu-umass_client-block [options*] &
```

Runs on:

QNX Neutrino

Options:

-l ***lun_options*[,*lun_options*]***

Specify options by logical unit, separated by commas; see "*LUN options* (p. 471)," below.

You can specify more than one -l. Specifying mutiple lun_options presents multiple logical devices to the host.

-n ***name***

The name associated with the resource manager (the default is `/dev/umass_client`). You must also specify the -r option.

-p ***prio***

Specify the event priority. The default is 20.

-r

Enable resource manager initialization. This provides access to the `_IO_DEVCTL` messages.

-s ***stack***

The name of the stack to attach to (default: `/dev/io-usb-dcd/io-usb`).

-U ***user_name***

-U ***uid*[:*gid*[,*sup_gid*]*]]**

Copyright © 2014, QNX Software Systems Limited

Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

-v

Be verbose. Additional v characters increase the verbosity (default I/O and memory errors):

- -v: Mass Storage Class (MSC) errors
- -vv: level debug info
- -vvv: `SCSI_WRITE10` debugging information
- -vvvv: Extra debugging information

-w *secs*

Wait *sec* seconds for the USB stack (default: 60 seconds).

LUN options

Each set of *lun_options* specified with the -l option is in this form:

`lun=`*`lun_id`*`[,`*`option`*`]*`

All options after a lun option target the preceeding specified LUN ID. LUN identifiers must start at index 0 and be sequential to a maximum of 3 (four LUNs total).

The options include:

fname=*filesystem*

Specify the filesystem name to be used by the driver. If not specified, no filesystem will be visible to the host.

iface=*if*

Specify the interface number associated with current option list (default=0). The interface specified should match a valid MSC interface number found in the device descriptors information.

ltype=*lun_type*

Copyright © 2014, QNX Software Systems Limited

Indicate the type of medimum presented to host. For CDROM devices, you must specify ltype=cd. You don't need to specifiy this option for other types of devices.

The cd option also implies the rdonly option. The provided filesystem would normally be a preformatted CDROM image.

rdonly

Mark the initial state of media as read-only.

scsi_pid=*string*

The SCSI product ID string (default: USB udisk).

scsi_rev=*string*

The SCSI revision string (default: 124).

scsi_vid=*string*

The SCSI vendor ID string (default: QNX Disk).

Use commas to separate the options. You must specify at least one option. Specify the lun, devno, or iface option first (when needed); otherwise values will be applied to default options and could cause unexpected behavior. The remaining options can be presented in any order.

Description:

The `devu-umass_client-block` driver is the function driver for USB Mass Storage Class Bulk-Only (BBB) devices. It also complies with the USB Mass Storage Class Bootability and the USB Mass Storage Class Compliance Test specifications. This driver, in combination with the USB device controller driver (DCD) stack (*`io-usb-dcd`* (p. 1018)) supports a MSC (Mass Storage Class) interface presented to a connected USB host.

The host machine (e.g., Windows, MAC, Linux) can't access the data stored in the filesystem (on the target hardware) at the same time as applications running on the target hardware. There will be a conflict as both filesystems believe they have exclusive access to the raw media for filesytem updates. Doing so will cause corruption of the filesystem as both filesytems may attempt to update the media without the other's knowledge. Once the board is disconnected or ejected from the host, it's safe to mount and access the filesystem from applications on the target hardware.

Copyright © 2014, QNX Software Systems Limited

Examples:

Start the driver without arguments. No filesystem is visible to the host:

```
devu-umass_client-block &
```

Start the driver and specify a filesystem associated with LUN 0:

```
devu-umass_client-block -l lun=0,fname=/tmp/disk.img &
```

Start the driver and specify a filesystem associated with LUNs 0 and 1, specifing that LUN 1 is a CDROM:

```
devu-umass_client-block -l lun=0,fname=/tmp/disk.img -l lun=1,ltype=cd,fname=/tmp/cdrom.img &
```

Do the same as the previous example, but using only one -l option to provide multiple LUN configurations. Increase the driver verbosity to level 3 (SCSI_WRITE10):

```
devu-umass_client-block -l lun=0,fname=/tmp/disk.img,lun=1,ltype=cd,fname=/tmp/cdrom.img -vvv &
```

Copyright © 2014, QNX Software Systems Limited

devu-xhci.so

Driver for Extensible Host Controller Interface (XHCI) USB controllers

You must be `root` to start this manager.

Syntax:

```
io-usb -d xhci [option[,option...]] ... &
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

ioport=*addr*

Register the base address of the USB controller. The default is to scan the PCI bus.

irq=*num*

The interrupt request number.

nosmm

Don't disable system management. The default is to disable it.

pindex=*num*

The instance of the controller on the PCI bus that you want to apply the arguments to.

prio=*num*

Set the pulse-handler thread's priority (default 21).

verbose=*num*

Set verbosity.

Copyright © 2014, QNX Software Systems Limited

Description:

The `devu-xhci.so` server provides support for computers that have Extensible Host Controller Interface (XHCI) USB controllers. This server creates the `/dev/io-usb/devu-xhci.so` device.

- If you specify arguments, you must include a *pindex* argument for all controllers you wish to use. The value of the *pindex* argument starts from 0.
- The `devu-xhci.so` stack supports high, full, and low speeds.

Examples:

Scan the PCI bus for all available controllers:

```
io-usb -dxhci
```

Attach to the first and third controller instances and set the *verbose* argument on the first controller instance as follows:

```
io-usb -dxhci pindex=0,verbose=4,pindex=2 &
```

Copyright © 2014, QNX Software Systems Limited

df

Report free disk space (POSIX)

Syntax:

```
df [-ghknP] [device|directory|file]*
```

Runs on:

QNX Neutrino

Options:

-g

Display all *statvfs()* information.

-h

Display the sizes in a "human-readable" form, using bytes, KB, MB, or GB as the units.

-k

Use 1024-byte units (the default is 512-byte).

-n

Display the filesystem mountpoints and types only.

-P

Display headings for the columns. (Output conforms to POSIX 1003.2/5.8.6.1 format.)

Description:

The `df` utility displays the amount of free disk space for the given devices, directories, and files.

By default, `df` reports its figures in 512-byte units. If you specify the -k option, it uses 1024-byte units; if you specify -h, it uses bytes.

Any block counts reported by the filesystem are rounded into 512- or 1024-byte units, and `df` always rounds down (i.e. to reflect whole blocks available). For a filesystem that doesn't use native 512-byte (or multiples thereof) blocks, this will result in round-off errors.

Copyright © 2014, QNX Software Systems Limited

Examples:

Display the sizes in 512-byte units:

```
$ df -P
Filesystem       512-blocks      Used Available Capacity  Mounted on
/dev/hd0t178       37190440   4378680  32811760      12%  /
/dev/hd0t177      122881088   4198452 118682636       4%  /fs/hd0-qnx6-2/
/dev/cd0                0         0         0     100%  (/fs/cd0/)
/dev/hd0          160086528 160086528         0     100%
```

Display the sizes in 1024-byte units:

```
$ df -kP
Filesystem      1024-blocks      Used Available Capacity  Mounted on
/dev/hd0t178       18595220   2189340  16405880      12%  /
/dev/hd0t177       61440544   2099226  59341318       4%  /fs/hd0-qnx6-2/
/dev/cd0                 0         0         0     100%  (/fs/cd0/)
/dev/hd0           80043264  80043264         0     100%
```

Display the sizes in bytes:

```
$ df -hP
Filesystem            Size      Used Available Capacity  Mounted on
/dev/hd0t178           18G      2.0G       16G      12%  /
/dev/hd0t177           59G      2.0G       57G       4%  /fs/hd0-qnx6-2/
/dev/cd0                 0         0         0     100%  (/fs/cd0/)
/dev/hd0                76G       76G         0     100%
```

Copyright © 2014, QNX Software Systems Limited

dhclient

Dynamic Host Configuration Protocol client

Syntax:

```
dhclient  [ -4 | -6 ] [ -S ] [ -N [ -N...  ] ] [ -T [ -T...
] ]
          [ -P [-P...  ] ] [ -p port ] [ -d ] [ -e VAR=value
]
          [ -q ] [ -1 ] [ -r | -x ] [ -lf lease-file ] [ -pf
pid-file ]
          [ -cf config-file ] [ -sf script-file ] [ -s server
 ]
         [ -g relay ] [ -n ] [ -nw ] [ -w ] [ -v ] [ --version
 ]
          [ if0 [ ...ifN ] ]
```

Runs on:

QNX Neutrino

Options:

-1

Cause `dhclient` to try once to get a lease. If it fails, `dhclient` exits with exit code 2. In DHCPv6, the -1 flag sets the maximum duration of the initial exchange to timeout (from `dhclient.conf`, default 60 seconds).

-4

Use the DHCPv4 protocol to obtain an IPv4 address and configuration parameters.

-6

Use the DHCPv6 protocol to obtain whatever IPv6 addresses are available along with configuration parameters. If you specify this option, the names of the default files include a “6”:

IPv4 name	IPv6 name
`dhclient.conf`	`dhclient6.conf`
`dhclient.leases`	`dhclient6.leases`
`dhclient.pid`	`dhclient6.pid`

-cf *config-file*

Copyright © 2014, QNX Software Systems Limited

The name of the file to read configuration information from. The default is `/etc/dhclient.conf` for DHCPv4, and `/etc/dhclient6.conf` for DHCPv6.

-d

Force `dhclient` to always run as a foreground process.

-e *VAR=value*

Define extra environment variables and their values.

-g *relay*

For testing purposes, set the *giaddr* field of all packets that the client sends to the given IP address.

-lf *lease-file*

The name of the lease database to use. The default is `/var/db/dhclient.leases` for DHCPv4, and `/var/db/dhclient6.leases` for DHCPv6.

-m

(QNX Neutrino extension) Write the resolver configuration to memory (using `confstr`) rather than to `/etc/resolv.conf`.

--no-resolve

(QNX Neutrino extension) Don't install the resolver configuration at all.

-N [-N...]

Restore the normal address query (which is disabled if you use temporary addresses (-T or prefix delegation (-P).

-n

Don't attempt to configure any interfaces. This is most likely to be useful in combination with the -w flag.

-nw

Make `dhclient` become a daemon immediately, rather than waiting until it has acquired an IP address.

-P [-P...]

Enable the IPv6 prefix delegation.

-p *port*

Copyright © 2014, QNX Software Systems Limited

The UDP port that the DHCP client should listen and transmit on; the default is port 68. This is mostly useful for debugging purposes.

-pf ***pid-file***

The name of the file in which `dhclient` stores its process ID. The default is `/var/run/dhclient.pid` for DHCPv4, and `/var/run/dhclient6.pid` for DHCPv6.

-q

Be quiet (although `dhclient` is quiet by default).

-r

Release the current lease. Once the lease has been released, the client exits.

-S

Use Information-request to get only (i.e., without address) stateless configuration parameters.

-s ***server***

The IP address or domain name to which `dhclient` should transmit any protocol messages that it sends before acquiring an IP address. The default is 255.255.255.255, the IP limited broadcast address. This is mostly for debugging purposes, and isn't supported by DHCPv6.

-sf ***script-file***

The name of the script file that configures the network interface. The default is `/sbin/dhclient-script`.

-T [-T...]

Ask for IPv6 temporary addresses, one set per -T flag.

-v

Make `dhclient` emit verbose messages displaying the startup sequence events until it has acquired an address.

-w

Make the DHCP client not exit if it isn't able to identify any network interfaces to configure. On laptop computers and other computers with hot-swappable I/O buses, it's possible that a broadcast interface may be added after system startup.

-x

Copyright © 2014, QNX Software Systems Limited

Tell any currently running client to exit gracefully without releasing leases first.

--version

Display the version number for `dhclient`, and then exit.

***if0* [*...ifN*]**

The names of the network interfaces that `dhclient` should attempt to configure. If you don't specify any interface names on the command line, `dhclient` identifies all network interfaces, eliminating non-broadcast interfaces if possible, and attempts to configure each interface.

Description:

The Internet Systems Consortium DHCP Client, `dhclient`, provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.

Operation

The DHCP protocol allows a host to contact a central server which maintains a list of IP addresses which may be assigned on one or more subnets. A DHCP client may request an address from this pool, and then use it on a temporary basis for communication on network. The DHCP protocol also provides a mechanism whereby a client can learn important details about the network to which it is attached, such as the location of a default router, the location of a name server, and so on.

If given the -4 command-line argument (default), `dhclient` will use the DHCPv4 protocol to obtain an IPv4 address and configuration parameters.

If given the -6 command-line argument, `dhclient` will use the DHCPv6 protocol to obtain whatever IPv6 addresses are available along with configuration parameters. But with -S, it uses Information-request to get only (i.e., without address) stateless configuration parameters.

The default DHCPv6 behavior is modified too with -T, which asks for IPv6 temporary addresses, one set per -T flag. -P enables the IPv6 prefix delegation. As temporary addresses or prefix delegation disables the normal address query, -N restores it. Note it is not recommended to mix queries of different types together, or even to share the lease file between them.

If given the --version command-line argument, `dhclient` will display its version number and exit.

On startup, `dhclient` reads the *`dhclient.conf`* (p. 492) file for configuration instructions. It then gets a list of all the network interfaces that are configured in the

Copyright © 2014, QNX Software Systems Limited

current system. For each interface, it attempts to configure the interface using the DHCP protocol.

In order to keep track of leases across system reboots and server restarts, `dhclient` keeps a list of leases it has been assigned in the *`dhclient.leases`* (p. 504) file. On startup, after reading the `dhclient.conf` file, `dhclient` reads the `dhclient.leases` file to refresh its memory about what leases it has been assigned.

When a new lease is acquired, it's appended to the end of the `dhclient.leases` file. In order to prevent the file from becoming arbitrarily large, from time to time `dhclient` creates a new `dhclient.leases` file from its in-core lease database. The old version of the `dhclient.leases` file is retained under the name `dhclient.leases~` until the next time `dhclient` rewrites the database.

Old leases are kept around in case the DHCP server is unavailable when `dhclient` is first invoked (generally during the initial system boot process). In that event, old leases from the `dhclient.leases` file which have not yet expired are tested, and if they are determined to be valid, they're used until either they expire or the DHCP server becomes available.

A mobile host which may sometimes need to access a network on which no DHCP server exists may be preloaded with a lease for a fixed address on that network. When all attempts to contact a DHCP server have failed, `dhclient` will try to validate the static lease, and if it succeeds,will use that lease until it is restarted.

A mobile host may also travel to some networks on which DHCP isn't available but BOOTP is. In that case, it may be advantageous to arrange with the network administrator for an entry on the BOOTP database, so that the host can boot quickly on that network rather than cycling through the list of old leases.

Command line

The names of the network interfaces that `dhclient` should attempt to configure may be specified on the command line. If no interface names are specified on the command line `dhclient` will normally identify all network interfaces, eliminating non-broadcast interfaces if possible, and attempt to configure each interface.

It's also possible to specify interfaces by name in the *`dhclient.conf`* (p. 492) file. If interfaces are specified in this way, then the client will only configure interfaces that are either specified in the configuration file or on the command line, and will ignore all other interfaces.

If the DHCP client should listen and transmit on a port other than the standard (port 68), the -p flag may used. It should be followed by the UDP port number that `dhclient` should use. This is mostly useful for debugging purposes. If a different port is specified for the client to listen on and transmit on, the client will also use a different destination port — one less than the specified port.

Copyright © 2014, QNX Software Systems Limited

The DHCP client normally transmits any protocol messages it sends before acquiring an IP address to 255.255.255.255, the IP limited broadcast address. For debugging purposes, it may be useful to have the server transmit these messages to some other address. This can be specified with the -s flag, followed by the IP address or domain name of the destination. This feature is not supported by DHCPv6.

For testing purposes, the *giaddr* field of all packets that the client sends can be set using the -g flag, followed by the IP address to send. This is only useful for testing, and should not be expected to work in any consistent or useful way.

The DHCP client will normally run in the foreground until it has configured an interface, and then will revert to running in the background. To force `dhclient` to always run as a foreground process, the -d flag should be specified. This is useful when running the client under a debugger, or when running it out of `inittab` on System V systems.

The `dhclient` daemon creates its own environment when executing the *`dhclient-script`* (p. 487) to do the grunt work of interface configuration. To define extra environment variables and their values, use the -e flag, followed by the environment variable name and value assignment, just as you'd assign a variable in a shell. For example, -e IF_METRIC=1.

The client normally prints no output during its startup sequence. It can be made to emit verbose messages displaying the startup sequence events until it has acquired an address by supplying the -v command-line argument. In either case, the client logs messages using the `syslog` facility. A -q command-line argument is provided for backwards compatibility, but since `dhclient` is quiet by default, it has no effect.

The client normally doesn't release the current lease as it is not required by the DHCP protocol. Some cable ISPs require their clients to notify the server if they wish to release an assigned IP address. The -r flag explicitly releases the current lease, and once the lease has been released, the client exits.

The -x flag tells any currently running client to exit gracefully without releasing leases first.

If the client is killed by a signal (for example at shutdown or reboot) it won't execute the `dhclient-script` at exit. However if you shut the client down gracefully with -r or -x, it will execute `dhclient-script` at shutdown with the specific reason for calling the script.

The -1 flag will cause `dhclient` to try once to get a lease. If it fails, `dhclient` exits with exit code 2. In DHCPv6, the -1 flag sets the maximum duration of the initial exchange to timeout (from `dhclient.conf`, default 60 seconds).

The DHCP client normally gets its configuration information from `/etc/dhclient.conf`, its lease database from `/var/db/dhclient.leases`, stores its process ID in a file called `/var/run/dhclient.pid`, and configures the network interface using `/sbin/dhclient-script`. To specify different names and/or locations for these files, use the -cf, -lf, -pf, and -sf flags, respectively, followed

Copyright © 2014, QNX Software Systems Limited

by the name of the file. This can be particularly useful if, for example, `/var/db` or `/var/run` hasn't yet been mounted when the DHCP client is started.

The DHCP client normally exits if it isn't able to identify any network interfaces to configure. On laptop computers and other computers with hot-swappable I/O buses, it's possible that a broadcast interface may be added after system startup. The -w flag can be used to cause the client not to exit when it doesn't find any such interfaces. The *omshell* (p. 1412) program can then be used to notify the client when a network interface has been added or removed, so that the client can attempt to configure an IP address on that interface.

The DHCP client can be directed not to attempt to configure any interfaces using the -n flag. This is most likely to be useful in combination with the -w flag.

The client can also be instructed to become a daemon immediately, rather than waiting until it has acquired an IP address. This can be done by supplying the -nw flag.

Configuration

The syntax of the *dhclient.conf* (p. 492) file is discussed separately.

OMAPI

The DHCP client provides some ability to control it while it is running, without stopping it. This capability is provided using OMAPI, an API for manipulating remote objects. OMAPI clients connect to the client using TCP/IP, authenticate, and can then examine the client's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the *dhcpctl*()* API or OMAPI itself. `Dhcpctl` is a wrapper that handles some of the housekeeping chores that OMAPI does not do automatically. Most things you'd want to do with the client can be done directly using the *omshell* (p. 1412) command, rather than having to write a special program.

The control object

The control object allows you to shut the client down, releasing all leases that it holds and deleting any DNS records it may have added. It also allows you to pause the client; this unconfigures any interfaces the client is using. You can then restart it, which causes it to reconfigure those interfaces. You would normally pause the client prior to going into hibernation or sleep on a laptop computer. You would then resume it after the power comes back. This allows PC cards to be shut down while the computer is hibernating or sleeping, and then reinitialized to their previous state once the computer comes out of hibernation or sleep.

The control object has one attribute: the state attribute. To shut the client down, set its state attribute to 2. It will automatically do a DHCPRELEASE. To pause it, set its state attribute to 3. To resume it, set its state attribute to 4.

Copyright © 2014, QNX Software Systems Limited

Files:

The `dhclient` depends on the following libraries and binaries:

- `libcrypto.so`
- `libsocket.so`
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)
- `/sbin/dhclient-script` (required; you can override it with -sf *script-file* on startup)

It uses the following configuration files:

/etc/dhclient6.conf **(p. 492)**

DHCP client configuration file for IPv6 (optional; defaults are used if this file isn't present; you can override it on startup).

/etc/dhclient.conf **(p. 492)**

DHCP client configuration file for IPv4.

dhclient6.leases **(p. 504)**

DHCP client lease database for IPv6 (optional; if not present, it will generate a new client ID on every startup; you can override it on startup).

dhclient.leases **(p. 504)**

DHCP client lease database for IPv4.

/sbin/dhclient-script **(p. 487)**

DHCP client network configuration script.

`/var/run/dhclient.pid`

Process ID of `dhclient`.

Contributing author:

`dhclient`(8) has been written for Internet Systems Consortium by Ted Lemon in cooperation with Vixie Enterprises. To learn more about Internet Systems Consortium, see *`http://www.isc.org`*. To learn more about Vixie Enterprises, see *`http://www.vix.com`*.

This client was substantially modified and enhanced by Elliot Poger for use on Linux while he was working on the MosquitoNet project at Stanford.

The current version owes much to Elliot's Linux enhancements, but was substantially reorganized and partially rewritten by Ted Lemon so as to use the same networking

Copyright © 2014, QNX Software Systems Limited

framework that the Internet Systems Consortium DHCP server uses. Much system-specific configuration code was moved into a shell script so that as support for more operating systems is added, it will not be necessary to port and maintain system-specific configuration code to these operating systems; instead, the shell script can invoke the native tools to accomplish the same purpose.

Copyright © 2014, QNX Software Systems Limited

dhclient-script

DHCP client network configuration script

Syntax:

```
dhclient-script
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The DHCP client network configuration script is invoked from time to time by *dhclient* (p. 478). This script is used by the DHCP client to set each interface's initial configuration prior to requesting an address, to test the address once it has been offered, and to set the interface's final configuration once a lease has been acquired. If no lease is acquired, the script is used to test predefined leases, if any, and also called once if no valid lease can be identified.

This script is not meant to be customized by the end user. If local customizations are needed, they should be possible using the enter and exit hooks provided (see "*Hooks* (p. 487)" for details). These hooks will allow the user to override the default behavior of the client in creating a `/etc/resolv.conf` file.

No standard client script exists for some operating systems, even though the actual client may work, so a pioneering user may well need to create a new script or modify an existing one. In general, customizations specific to a particular computer should be done in the */etc/dhclient.conf* (p. 492) file. If you find that you can't make such a customization without customizing `/etc/dhclient.conf` or using the enter and exit hooks, please submit a bug report.

Hooks

When it starts, the client script first defines a shell function, `make_resolv_conf`, which is later used to create the `/etc/resolv.conf` file. To override the default behavior, redefine this function in the enter hook script.

After defining the `make_resolv_conf` function, the client script checks for the presence of an executable `/etc/dhclient-enter-hooks` script, and if present, it invokes the script inline, using the Bourne shell "dot" (`.`) command. The entire environment documented under "*Operations* (p. 488)" is available to this script, which

Copyright © 2014, QNX Software Systems Limited

may modify the environment if needed to change the behavior of the script. If an error occurs during the execution of the script, it can set the *exit_status* variable to a nonzero value, and `/sbin/dhclient-script` will exit with that error code immediately after the client script exits.

After all processing has completed, `/sbin/dhclient-script` checks for the presence of an executable `/etc/dhclient-exit-hooks` script, which if present is invoked using the "dot" command. The exit status of `dhclient-script` is passed to `dhclient-exit-hooks` in the *exit_status* shell variable, and is always zero if the script succeeded at the task for which it was invoked. The rest of the environment as described previously for `dhclient-enter-hooks` is also present. The `/etc/dhclient-exit-hooks` script can modify the value of *exit_status* to change the exit status of `dhclient-script`.

Operations

When `dhclient` needs to invoke the client configuration script, it defines a set of variables in the environment, and then invokes `/sbin/dhclient-script`. In all cases, *$reason* is set to the name of the reason why the script has been invoked. The following reasons are currently defined:

`MEDIUM`

The DHCP client is requesting that an interface's media type be set. The interface name is passed in *$interface*, and the media type is passed in *$medium*.

`PREINIT`

The DHCP client is requesting that an interface be configured as required in order to send packets prior to receiving an actual address. For clients which use the BSD socket library, this means configuring the interface with an IP address of 0.0.0.0 and a broadcast address of 255.255.255.255. For other clients, it may be possible to simply configure the interface without actually giving it an IP address at all. The interface name is passed in *$interface*, and the media type in *$medium*.

If an IP alias has been declared in `dhclient.conf`, its address will be passed in *$alias_ip_address*, and that IP alias should be deleted from the interface, along with any routes to it.

`BOUND`

The DHCP client has done an initial binding to a new address. The new IP address is passed in *$new_ip_address*, and the interface name is passed in *$interface*. The media type is passed in *$medium*. Any options acquired from the server are passed using the option name described in the *DHCP options* (p. 514) entry, except that dashes (`-`) are replaced by underscores

Copyright © 2014, QNX Software Systems Limited

(_) in order to make valid shell variables, and the variable names start with new_. So for example, the new subnet mask would be passed in *$new_subnet_mask*.

Before actually configuring the address, dhclient-script should somehow ARP for it and exit with a nonzero status if it receives a reply. In this case, the client will send a DHCPDECLINE message to the server and acquire a different address. This may also be done in the RENEW, REBIND, or REBOOT states, but isn't required, and indeed may not be desirable.

When a binding has been completed, a lot of network parameters are likely to need to be set up. A new /etc/resolv.conf needs to be created, using the values of *$new_domain_name* and *$new_domain_name_servers* (which may list more than one server, separated by spaces). A default route should be set using *$new_routers*, and static routes may need to be set up using *$new_static_routes*.

If an IP alias has been declared, it must be set up here. The alias IP address will be written as *$alias_ip_address*, and other DHCP options that are set for the alias (e.g., subnet mask) will be passed in variables named as described previously except starting with $alias_ instead of $new_. Care should be taken that the alias IP address not be used if it's identical to the bound IP address (*$new_ip_address*), since the other alias parameters may be incorrect in this case.

RENEW

When a binding has been renewed, the script is called as in BOUND, except that in addition to all the variables starting with $new_, there is another set of variables starting with $old_. Persistent settings that may have changed need to be deleted; for example, if a local route to the bound address is being configured, the old local route should be deleted. If the default route has changed, the old default route should be deleted. If the static routes have changed, the old ones should be deleted. Otherwise, processing can be done as with BOUND.

REBIND

The DHCP client has rebound to a new DHCP server. This can be handled as with RENEW, except that if the IP address has changed, the ARP table should be cleared.

REBOOT

The DHCP client has successfully reacquired its old address after a reboot. This can be processed as with BOUND.

EXPIRE

Copyright © 2014, QNX Software Systems Limited

The DHCP client has failed to renew its lease or acquire a new one, and the lease has expired. The IP address must be relinquished, and all related parameters should be deleted, as in `RENEW` and `REBIND`.

`FAIL`

The DHCP client has been unable to contact any DHCP servers, and any leases that have been tested have not proved to be valid. The parameters from the last lease tested should be deconfigured. This can be handled in the same way as `EXPIRE`.

`STOP`

The `dhclient` has been told to shut down gracefully; the `dhclient-script` should unconfigure or shut down the interface as appropriate.

`RELEASE`

The `dhclient` has been executed using the -r flag, indicating that the administrator wishes it to release its lease(s); `dhclient-script` should unconfigure or shut down the interface.

`NBI`

No-Broadcast-Interfaces. The `dhclient` was unable to find any interfaces on which it believed it should commence DHCP. What `dhclient-script` should do in this situation is entirely up to the implementer.

`TIMEOUT`

The DHCP client has been unable to contact any DHCP servers. However, an old lease has been identified, and its parameters have been passed in as with `BOUND`. The client configuration script should test these parameters and, if it has reason to believe they are valid, should exit with a value of zero. If not, it should exit with a nonzero value.

The usual way to test a lease is to set up the network as with `REBIND` (since this may be called to test more than one lease) and then ping the first router defined in *$routers*. If a response is received, the lease must be valid for the network to which the interface is currently connected. It would be more complete to try to ping all of the routers listed in *$new_routers*, as well as those listed in *$new_static_routes*, but current scripts do not do this.

Files:

The `dhclient-script` depends on the following libraries and binaries:

- `ifconfig` (which needs `io-pkt-*` and `libsocket.so`)

Copyright © 2014, QNX Software Systems Limited

- sh
- route
- hostname
- getconf
- setconf
- cat
- mv
- If /etc/dhclient-enter-hooks and/or /etc/dhclient-exit-hooks exists, then dhclient-script will run them too.

It uses the following configuration files:

- /etc/resolv.conf (optional; needed to update the DNS server; must be read/write)

Contributing author:

dhclient-script has been written for Internet Systems Consortium by Ted Lemon in cooperation with Vixie Enterprises. To learn more about Internet Systems Consortium, see *http://www.isc.org*. To learn more about Vixie Enterprises, see *http://www.vix.com*.

Caveats:

If more than one interface is being used, there's no obvious way to avoid clashes between server-supplied configuration parameters. For example, the stock dhclient-script rewrites /etc/resolv.conf. If more than one interface is being configured, /etc/resolv.conf will be repeatedly initialized to the values provided by one server, and then the other. Assuming the information provided by both servers is valid, this shouldn't cause any real problems, but it could be confusing.

Copyright © 2014, QNX Software Systems Limited

dhclient.conf, dhclient6.conf

DHCP client configuration file

Name:

- `/etc/dhclient.conf` for DHCPv4
- `/etc/dhclient6.conf` for DHCPv6

Description:

The `dhclient.conf` file contains configuration information for *`dhclient`* (p. 478), the Internet Systems Consortium DHCP Client.

The `dhclient.conf` file is a free-form ASCII text file. It's parsed by the recursive-descent parser built into `dhclient`. The file may contain extra tabs and newlines for formatting purposes. Keywords in the file are case-insensitive. Comments may be placed anywhere within the file (except within quotes). Comments begin with the # character and end at the end of the line.

The `dhclient.conf` file can be used to configure the behavior of the client in a wide variety of ways: protocol timing, information requested from the server, information required of the server, defaults to use if the server doesn't provide certain information, values with which to override information provided by the server, or values to prepend or append to information provided by the server. The configuration file can also be preinitialized with addresses to use on networks that don't have DHCP servers.

Protocol timing

The timing behavior of the client needn't be configured by the user. If no timing configuration is provided by the user, a fairly reasonable timing behavior will be used by default, one which results in fairly timely updates without placing an inordinate load on the server.

The following statements can be used to adjust the timing behavior of the DHCP client if required, however:

- The `timeout` statement:

  ```
  timeout time ;
  ```

  The `timeout` statement determines the amount of time that must pass between the time that the client begins to try to determine its address and the time that it decides that it's not going to be able to contact a server. By default, this timeout is sixty seconds. After the timeout has passed, if there are any static leases defined in the configuration file, or any leases remaining in the lease database that have not yet expired, the client will loop through these leases attempting to validate

Copyright © 2014, QNX Software Systems Limited

them, and if it finds one that appears to be valid, it will use that lease's address. If there are no valid static leases or unexpired leases in the lease database, the client will restart the protocol after the defined retry interval.

- The `retry` statement:

  ```
  retry time ;
  ```

  The `retry` statement determines the time that must pass after the client has determined that there is no DHCP server present before it tries again to contact a DHCP server. By default, this is five minutes.

- The `select-timeout` statement:

  ```
  select-timeout time ;
  ```

  It's possible (some might say desirable) for there to be more than one DHCP server serving any given network. In this case, it's possible that a client may be sent more than one offer in response to its initial lease discovery message. It may be that one of these offers is preferable to the other (e.g., one offer may have the address the client previously used, and the other may not).

  The `select-timeout` is the time after the client sends its first lease discovery request at which it stops waiting for offers from servers, assuming that it has received at least one such offer. If no offers have been received by the time the `select-timeout` has expired, the client will accept the first offer that arrives.

  By default, the `select-timeout` is zero seconds; that is, the client will take the first offer it sees.

- The `reboot` statement:

  ```
  reboot time ;
  ```

  When the client is restarted, it first tries to reacquire the last address it had. This is called the INIT-REBOOT state. If it is still attached to the same network it was attached to when it last ran, this is the quickest way to get started. The `reboot` statement sets the time that must elapse after the client first tries to reacquire its old address before it gives up and tries to discover a new address. By default, the reboot timeout is ten seconds.

- The `backoff-cutoff` statement:

  ```
  backoff-cutoff time ;
  ```

  The client uses an exponential backoff algorithm with some randomness, so that if many clients try to configure themselves at the same time, they will not make their requests in lockstep. The `backoff-cutoff` statement determines the maximum amount of time that the client is allowed to back off; the actual value

Copyright © 2014, QNX Software Systems Limited

will be evaluated randomly between 1/2 to 1 1/2 times the time specified. It defaults to two minutes.

- The `initial-interval` statement:

  ```
  initial-interval time ;
  ```

  The `initial-interval` statement sets the amount of time between the first attempt to reach a server and the second attempt to reach a server. Each time a message is sent, the interval between messages is incremented by twice the current interval multiplied by a random number between zero and one. If it is greater than the backoff-cutoff amount, it is set to that amount. It defaults to ten seconds.

Lease requirements and requests

The DHCP protocol allows the client to request that the server send it specific information, and not send it other information that it is not prepared to accept. The protocol also allows the client to reject offers from servers if they don't contain information the client needs, or if the information provided isn't satisfactory.

There's a variety of data contained in offers that DHCP servers send to DHCP clients. The data that can be specifically requested is what are called *DHCP options*. For more information, see the DHCP Options chapter in this guide.

- The `request` statement:

  ```
  [ also ] request [ [ option-space . ] option ] [, ... ];
  ```

  The `request` statement causes the client to request that any server responding to the client send the client its values for the specified options. Only the option names — not option parameters — should be specified in the `request` statement. By default, the DHCP server requests the `subnet-mask`, `broadcast-address`, `time-offset`, `routers`, `domain-name`, `domain-name-servers`, and `host-name` options.

  If you enter a `request` statement, you override this default, and these options will not be requested.

  In some cases, it may be desirable to send no parameter request list at all. To do this, simply write the request statement, but specify no parameters:

  ```
  request;
  ```

  In most cases, it's desirable to simply add one option to the request list which is of interest to the client in question. In this case, it is best to `also request` the additional options:

  ```
  also request domain-search, dhcp6.sip-servers-addresses;
  ```


Copyright © 2014, QNX Software Systems Limited

- The `require` statement:

  ```
  [ also ] require [ [ option-space . ] option ] [, ... ];
  ```

  The `require` statement lists options that must be sent in order for an offer to be accepted. Offers that don't contain all the listed options will be ignored. There's no default `require` list.

  ```
  require name-servers;

  interface eth0 {
      also require domain-search;
  }
  ```

- The `send` statement:

  ```
   send { [ option declaration ]
  [, ... option declaration ]}
  ```

  The `send` statement causes the client to send the specified options to the server with the specified values. These are full option declarations as described in the DHCP Options chapter. Options that are always sent in the DHCP protocol shouldn't be specified here, except that the client can specify a `requested-lease-time` option other than the default requested lease time, which is two hours. The other obvious use for this statement is to send information to the server that will allow it to differentiate between this client and other clients or kinds of clients.

DHCP operation

The client doesn't yet have a default DHCPv6 Option Request Option (ORO), nor has it been integrated with the `request` and `require` syntax above. It is necessary to configure an ORO then:

```
send dhcp6.oro 1, 2, 7, 12, 13, 23, 24, 39;
```

The above ORO will request both identifiers (server, client), the preference, unicast, nameservers, domain-search, and FQDN(v6) options.

Dynamic DNS

The client now has some very limited support for doing DNS updates when a lease is acquired. This is prototypical, and probably doesn't do what you want. It also only works if you happen to have control over your DNS server, which isn't very likely.

Everything in this section is true whether you are using DHCPv4 or DHCPv6. The exact same syntax is used for both.

To make it work, you have to declare a key and zone as in the DHCP server (see *dhcpd.conf* (p. 566) for details). You also need to configure the `fqdn` option on the client, as follows:

```
send fqdn.fqdn "grosse.fugue.com.";
```

Copyright © 2014, QNX Software Systems Limited

```
send fqdn.encoded on;
send fqdn.server-update off;
also request fqdn, dhcp6.fqdn;
```

The `fqdn.fqdn` option *must* be a fully-qualified domain name. You *must* define a zone statement for the zone to be updated. The `fqdn.encoded` option may need to be set to `on` or `off`, depending on the DHCP server you're using.

- The `do-forward-updates` statement:

  ```
  do-forward-updates [ flag ];
  ```

  If you want to do DNS updates in the DHCP client script (see *dhclient-script* (p. 487)) rather than having the DHCP client do the update directly (for example, if you want to use SIG(0) authentication, which isn't supported directly by the DHCP client, you can instruct the client not to do the update using the `do-forward-updates` statement. The *flag* should be `true` if you want the DHCP client to do the update, and `false` if you don't want the DHCP client to do the update. By default, the DHCP client will do the DNS update.

Option modifiers

In some cases, a client may receive option data from the server which isn't really appropriate for that client, or may not receive information that it needs, and for which a useful default value exists. It may also receive information which is useful, but which needs to be supplemented with local information. To handle these needs, several option modifiers are available:

- The `default` statement:

  ```
  default [ option declaration ] ;
  ```

  If for some option, the client should use the value supplied by the server, but needs to use some default value if no value was supplied by the server, these values can be defined in the `default` statement.

- The `supersede` statement:

  ```
  supersede [ option declaration ] ;
  ```

  If for some option the client should always use a locally-configured value or values rather than whatever is supplied by the server, these values can be defined in the `supersede` statement.

- The `prepend` statement:

  ```
  prepend [ option declaration ] ;
  ```

  If for some set of options the client should use a value you supply, and then use the values supplied by the server, if any, these values can be defined in the `prepend` statement. The `prepend` statement can be used only for options that

Copyright © 2014, QNX Software Systems Limited

allow more than one value to be given. This restriction isn't enforced; if you ignore it, the behavior will be unpredictable.

- The append statement:

```
append [ option declaration ] ;
```

If for some set of options the client should first use the values supplied by the server, if any, and then use values you supply, these values can be defined in the append statement. The append statement can be used only for options that allow more than one value to be given. This restriction isn't enforced; if you ignore it, the behavior will be unpredictable.

Lease declaration

The lease declaration is as follows:

```
lease { lease-declaration [ ... lease-declaration ] }
```

The DHCP client may decide after some period of time (see "*Protocol timing* (p. 492)") that it isn't going to succeed in contacting a server. At that time, it consults its own database of old leases and tests each one that hasn't yet timed out by pinging the listed router for that lease to see if that lease could work. It is possible to define one or more *fixed* leases in the client configuration file for networks where there's no DHCP or BOOTP service, so that the client can still automatically configure its address. This is done with the lease statement.

The lease statement is also used in the *dhclient.leases* (p. 504) file in order to record leases that have been received from DHCP servers. Some of the syntax for leases as described below is needed only in the dhclient.leases file. Such syntax is documented here for completeness.

A lease statement consists of the lease keyword, followed by a left curly brace, followed by one or more lease declaration statements, followed by a right curly brace. The following lease declarations are possible:

bootp;

The bootp statement is used to indicate that the lease was acquired using the BOOTP protocol rather than the DHCP protocol. It is never necessary to specify this in the client configuration file. The client uses this syntax in its lease database file.

interface "*string*";

The interface lease statement is used to indicate the interface on which the lease is valid. If set, this lease will be tried only on a particular interface. When the client receives a lease from a server, it always records the interface number on which it received that lease. If predefined leases are specified

Copyright © 2014, QNX Software Systems Limited

in the `dhclient.conf` file, the interface should also be specified, although this isn't required.

`fixed-address` *ip-address*;

The `fixed-address` statement is used to set the IP address of a particular lease. This is required for all lease statements. The IP address must be specified as a dotted quad (e.g., 12.34.56.78).

`filename` "*string*";

The `filename` statement specifies the name of the boot filename to use. This isn't used by the standard client configuration script, but is included for completeness.

`server-name` "*string*";

The `server-name` statement specifies the name of the boot server name to use. This is also not used by the standard client configuration script.

`option` *option-declaration*;

The `option` statement is used to specify the value of an option supplied by the server, or, in the case of predefined leases declared in `dhclient.conf`, the value that the user wishes the client configuration script to use if the predefined lease is used.

`script` "*script-name*";

The `script` statement is used to specify the pathname of the DHCP client configuration script. This script is used by the DHCP client to set each interface's initial configuration prior to requesting an address, to test the address once it has been offered, and to set the interface's final configuration once a lease has been acquired. If no lease is acquired, the script is used to test predefined leases, if any, and also called once if no valid lease can be identified. For more information, see *`dhclient-script`* (p. 487).

`vendor option space` "*name*";

The `vendor option space` statement is used to specify which option space should be used for decoding the vendor-encapsulate-options option if one is received. The `dhcp-vendor-identifier` can be used to request a specific class of vendor options from the server. See the entry for *DHCP options* (p. 514) for details.

`medium` "*media setup*";

Copyright © 2014, QNX Software Systems Limited

The `medium` statement can be used on systems where network interfaces can't automatically determine the type of network to which they are connected. The *media setup* string is a system-dependent parameter which is passed to the DHCP client configuration script when initializing the interface. On Unix and Unix-like systems, the argument is passed on the *ifconfig* (p. 957) command line when configuring the interface.

The DHCP client automatically declares this parameter if it uses a media type (see the `media` statement) when configuring the interface in order to obtain a lease. This statement should be used in predefined leases only if the network interface requires media type configuration.

`renew` *date*;

`rebind` *date*;

`expire` *date*;

The `renew` statement defines the time at which the DHCP client should begin trying to contact its server to renew a lease that it's using. The `rebind` statement defines the time at which the DHCP client should begin to try to contact *any* DHCP server in order to renew its lease. The `expire` statement defines the time at which the DHCP client must stop using a lease if it hasn't been able to contact a server in order to renew it.

These declarations are automatically set in leases acquired by the DHCP client, but must also be configured in predefined leases; a predefined lease whose expiry time has passed will not be used by the DHCP client.

Dates are specified in one of the following ways. The software will output times in these formats depending on the value of the the `db-time-format` configuration parameter:

- `default` — date values appear as follows:

  weekday year/*month*/*day hour*:*minute*:*second*

  The weekday is present to make it easy for a human to tell when a lease expires; it's specified as a number from zero to six, with zero being Sunday. When you're declaring a predefined lease, you can always specify it as zero. The year is specified with the century, so it should generally be four digits except for really long leases. The month is specified as a number starting with 1 for January. The day of the month is likewise specified starting with 1. The hour is a number between 0 and 23, the minute a number between 0 and 59, and the second also a number between 0 and 59.

- `local` — date values appear as follows:

Copyright © 2014, QNX Software Systems Limited

epoch *seconds-since-epoch*; # *day-name month-name day-number hours*:*minutes*:*seconds year*

The *seconds-since-epoch* is as according to the system's local clock (often referred to as "Unix time"). The # symbol supplies a comment that describes what actual time this is as according to the system's configured time zone, at the time the value was written. It is provided only for human inspection; the epoch time is the only recommended value for machine inspection.

Note that when defining a static lease, you may use either time format you wish, and you needn't include the comment or values after it.

If the time is infinite in duration, then the *date* is `never` instead of an actual date.

Alias declarations

```
alias {  declarations ... }
```

Some DHCP clients running TCP/IP roaming protocols may require that in addition to the lease they may acquire via DHCP, their interface also be configured with a predefined IP alias so that they can have a permanent IP address even while roaming. The Internet Systems Consortium DHCP client doesn't support roaming with fixed addresses directly, but in order to facilitate such experimentation, the dhcp client can be set up to configure an IP alias using the `alias` declaration.

The alias declaration resembles a lease declaration, except that options other than the subnet-mask option are ignored by the standard client configuration script, and expiry times are ignored. A typical alias declaration includes an interface declaration, a fixed-address declaration for the IP alias address, and a subnet-mask option declaration. A medium statement should never be included in an alias declaration.

Other declarations

`db-time-format [ default | local ];`

The `db-time-format` option determines which output methods is used for printing times in leases files. The `default` format provides day-and-time in UTC, whereas `local` uses seconds-since-epoch to store the time value, and helpfully places a local time zone time in a comment on the same line. The formats are described in detail in "*Lease declarations* (p. 497)," above.

`reject` *cidr-ip-address* [, ... *cidr-ip-address*];

The `reject` statement causes the DHCP client to reject offers from servers whose server identifier matches any of the specified hosts or subnets. This can be used to avoid being configured by rogue or misconfigured dhcp

Copyright © 2014, QNX Software Systems Limited

servers, although it should be a last resort — better to track down the bad DHCP server and fix it.

The *cidr-ip-address* configuration type is of the form *ip-address*[*/prefixlen*], where *ip-address* is a dotted quad IP address, and *prefixlen* is the CIDR prefix length of the subnet, counting the number of significant bits in the netmask starting from the leftmost end. For example:

```
reject 192.168.0.0/16, 10.0.0.5;
```

The above example would cause offers from any server identifier in the entire RFC 1918 “Class C” network 192.168.0.0/16, or the specific single address 10.0.0.5, to be rejected.

`interface` "*name*" { *declarations* ... }

A client with more than one network interface may require different behavior, depending on which interface is being configured. All timing parameters and declarations other than lease and alias declarations can be enclosed in an interface declaration, and those parameters will then be used only for the interface that matches the specified name. Interfaces for which there is no interface declaration will use the parameters declared outside of any interface declaration, or the default settings.

ISC `dhclient` maintains only one list of interfaces, which is either determined at startup from command-line arguments, or otherwise is autodetected. If you supplied the list of interfaces on the command line, this configuration clause will add the named interface to the list in such a way that will cause it to be configured by DHCP, which may not be the result you had intended. This is an undesirable side effect that will be addressed in a future release.

`pseudo` "*name*" *real-name* { *declarations* ... }

Under some circumstances it can be useful to declare a pseudo-interface and have the DHCP client acquire a configuration for that interface. Each interface that the DHCP client is supporting normally has a DHCP client state machine running on it to acquire and maintain its lease. A pseudo-interface is just another state machine running on the interface named *real-name*, with its own lease and its own state. If you use this feature, you must provide a client identifier for both the pseudo-interface and the actual interface, and the two identifiers must be different. You must also

Copyright © 2014, QNX Software Systems Limited

provide a separate client script for the pseudo-interface to do what you want with the IP address. For example:

```
interface "ep0" {
     send dhcp-client-identifier "my-client-ep0";
}
pseudo "secondary" "ep0" {
     send dhcp-client-identifier "my-client-ep0-secondary";
     script "/etc/dhclient-secondary";
}
```

The client script for the pseudo-interface shouldn't configure the interface up or down; essentially, all it needs to handle are the states where a lease has been acquired or renewed, and the states where a lease has expired. See *dhclient-script* (p. 487) for more information.

media "*media setup*" [, "*media setup*", ...];

The media statement defines one or more media configuration parameters which may be tried while attempting to acquire an IP address. The DHCP client will cycle through each media setup string on the list, configuring the interface using that setup and attempting to boot, and then trying the next one. This can be used for network interfaces which aren't capable of sensing the media type unaided; whichever media type succeeds in getting a request to the server and hearing the reply is probably right (no guarantees).

The media setup is only used for the initial phase of address acquisition (the DHCPDISCOVER and DHCPOFFER packets). Once an address has been acquired, the DHCP client will record it in its lease database and will record the media type used to acquire the address. Whenever the client tries to renew the lease, it will use that same media type. The lease must expire before the client will go back to cycling through media types.

Contributing author

dhclient was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Information about Internet Systems Consortium can be found at *http://www.isc.org*.

See also:

RFC2132, *RFC2131*

Examples:

The following configuration file is used on a laptop running NetBSD 1.3. The laptop has an IP alias of 192.5.5.213, and has one interface, ep0 (a 3com 3C589C). Booting intervals have been shortened somewhat from the default, because the client is known

Copyright © 2014, QNX Software Systems Limited

to spend most of its time on networks with little DHCP activity. The laptop does roam to multiple networks.

```
timeout 60;
retry 60;
reboot 10;
select-timeout 5;
initial-interval 2;
reject 192.33.137.209;

interface "ep0" {
    send host-name "andare.fugue.com";
    send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
    send dhcp-lease-time 3600;
    supersede domain-name "fugue.com rc.vix.com home.vix.com";
    prepend domain-name-servers 127.0.0.1;
    request subnet-mask, broadcast-address, time-offset, routers,
         domain-name, domain-name-servers, host-name;
    require subnet-mask, domain-name-servers;
    script "/sbin/dhclient-script";
    media "media 10baseT/UTP", "media 10base2/BNC";
}

alias {
  interface "ep0";
  fixed-address 192.5.5.213;
  option subnet-mask 255.255.255.255;
}
```

This is a very complicated `dhclient.conf` file; in general, yours should be much simpler. In many cases, it's sufficient to just create an empty `dhclient.conf` file; the defaults are usually fine.

Copyright © 2014, QNX Software Systems Limited

dhclient.leases, dhclient6.leases

DHCP client database of acquired leases

Name:

- `/var/db/dhclient.leases` for DHCPv4
- `/var/db/dhclient6.leases` for DHCPv6

Description:

The Internet Systems Consortium DHCP client keeps a persistent database of leases that it has acquired that are still valid. The database is a free-form ASCII file containing one valid declaration per lease. If more than one declaration appears for a given lease, the last one in the file is used. The file is written as a log, so this is not an unusual occurrence.

The format of the lease declarations is described in the entry for *dhclient.conf* (p. 492).

Contributing author:

`dhclient` was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Information about Internet Systems Consortium can be found at *http://www.isc.org*.

See also:

RFC2132, *RFC2131*

Copyright © 2014, QNX Software Systems Limited

DHCP Conditional Evaluation

Specify conditional behavior for DHCP servers and clients

Description:

The Internet Systems Consortium DHCP client and server both provide the ability to perform conditional behavior depending on the contents of packets they receive. This chapter describes the syntax for specifying this conditional behavior.

Conditional behavior

Conditional behavior is specified using the `if` statement and the `else` or `elsif` statements. A conditional statement can appear anywhere that a regular statement (e.g., an option statement) can appear, and can enclose one or more such statements. A typical conditional statement in a server might be:

```
if option dhcp-user-class = "accounting" {
  max-lease-time 17600;
  option domain-name "accounting.example.org";
  option domain-name-servers ns1.accounting.example.org,
                    ns2.accounting.example.org;
} elsif option dhcp-user-class = "sales" {
  max-lease-time 17600;
  option domain-name "sales.example.org";
  option domain-name-servers ns1.sales.example.org,
                    ns2.sales.example.org;
} elsif option dhcp-user-class = "engineering" {
  max-lease-time 17600;
  option domain-name "engineering.example.org";
  option domain-name-servers ns1.engineering.example.org,
                    ns2.engineering.example.org;
} else {
  max-lease-time 600;
  option domain-name "misc.example.org";
  option domain-name-servers ns1.misc.example.org,
                    ns2.misc.example.org;
}
```

On the client side, an example of conditional evaluation might be:

```
# example.org filters DNS at its firewall, so we have to use their DNS
# servers when we connect to their network.   If we are not at
# example.org, prefer our own DNS server.
if not option domain-name = "example.org" {
  prepend domain-name-servers 127.0.0.1;
}
```

The `if` statement and the `elsif` continuation statement both take boolean expressions as their arguments. That is, they take expressions that, when evaluated, produce a boolean result. If the expression evaluates to true, then the statements enclosed in braces following the `if` statement are executed, and all subsequent `elsif` and `else` clauses are skipped. Otherwise, each subsequent `elsif` clause's expression is checked, until an `elsif` clause is encountered whose test evaluates to true. If such a clause is found, the statements in braces following it are executed, and then any subsequent `elsif` and `else` clauses are skipped. If all the `if` and `elsif` clauses are checked, but none of their expressions evaluate true, then if there's an `else` clause, the

Copyright © 2014, QNX Software Systems Limited

statements enclosed in braces following the `else` are evaluated. Boolean expressions that evaluate to null are treated as false in conditionals.

Boolean expressions

The DHCP distribution supports the following boolean expressions:

data-expression-1 = data-expression-2

The = operator compares the values of two data expressions, returning true if they are the same, false if they are not. If either the left-hand side or the right-hand side are null, the result is also null.

data-expression-1 ~= data-expression-2 data-expression-1 ~~ data-expression-2

The ~= and ~~ operators (not available on all systems) perform extended `regex` matching of the values of two data expressions, returning true if *data-expression-1* matches against the regular expression evaluated by *data-expression-2*, or false if it does not match or encounters some error. If either the left-hand side or the right-hand side is null, the result is also false. The ~~ operator differs from the ~= operator in that it is case-insensitive.

boolean-expression-1* `and` *boolean-expression-2

The `and` operator evaluates to true if the boolean expression on the left-hand side and the boolean expression on the right-hand side both evaluate to true. Otherwise, it evaluates to false. If either the expression on the left-hand side or the expression on the right-hand side is null, the result is null.

boolean-expression-1* `or` *boolean-expression-2

The `or` operator evaluates to true if either the boolean expression on the left-hand side or the boolean expression on the right-hand side evaluate to true. Otherwise, it evaluates to false. If either the expression on the left-hand side or the expression on the right-hand side is null, the result is null.

`not` *boolean-expression*

The `not` operator evaluates to true if *boolean-expression* evaluates to false, and returns false if *boolean-expression* evaluates to true. If *boolean-expression* evaluates to null, the result is also null.

`exists` *option-name*

The `exists` expression returns true if the specified option exists in the incoming DHCP packet being processed.

known

Copyright © 2014, QNX Software Systems Limited

The `known` expression returns true if the client whose request is currently being processed is known — that is, if there's a host declaration for it.

`static`

The `static` expression returns true if the lease assigned to the client whose request is currently being processed is derived from a static address assignment.

Data expressions

Several of the boolean expressions above depend on the results of evaluating data expressions. A list of these expressions is provided here.

`substring` (*data-expr*, *offset*, *length*)

The `substring` operator evaluates the data expression and returns the substring of the result of that evaluation that starts *offset* bytes from the beginning, continuing for *length* bytes. The *offset* and *length* are both numeric expressions. If *data-expr*, *offset* or *length* evaluates to null, then the result is also null. If *offset* is greater than or equal to the length of the evaluated data, then a zero-length data string is returned. If *length* is greater than the remaining *length* of the evaluated data after *offset*, then a data string containing all data from *offset* to the end of the evaluated data is returned.

`suffix` (*data-expr*, *length*)

The `suffix` operator evaluates *data-expr* and returns the last *length* bytes of the result of that evaluation. The *length* is a numeric expression. If *data-expr* or *length* evaluates to null, then the result is also null. If `suffix` evaluates to a number greater than the length of the evaluated data, then the evaluated data is returned.

`lcase` (*data-expr*)

The `lcase` function returns the result of evaluating *data-expr* converted to lower case. If *data-expr* evaluates to null, then the result is also null.

`ucase` (*data-expr*)

The `ucase` function returns the result of evaluating *data-expr* converted to upper case. If *data-expr* evaluates to null, then the result is also null.

`option` *option-name*

The `option` operator returns the contents of the specified option in the packet to which the server is responding.

Copyright © 2014, QNX Software Systems Limited

`config-option` *option-name*

The `config-option` operator returns the value for the specified option that the DHCP client or server has been configured to send.

`hardware`

The `hardware` operator returns a data string whose first element is the type of network interface indicated in packet being considered, and whose subsequent elements are client's link-layer address. If there is no packet, or if the *RFC2131* `hlen` field is invalid, then the result is null. Hardware types include ethernet (1), token-ring (6), and fddi (8). Hardware types are specified by the IETF, and details on how the type numbers are defined can be found in RFC2131 (in the ISC DHCP distribution, this is included in the `doc/` subdirectory).

`packet` (*offset*, *length*)

The `packet` operator returns the specified portion of the packet being considered, or null in contexts where no packet is being considered. The *offset* and *length* are applied to the contents of the packet as in the `sub string` operator.

string

A string, enclosed in quotes, may be specified as a data expression, and returns the text between the quotes, encoded in ASCII. The backslash (`\`) character is treated specially, as in C programming: `\t` means TAB, `\r` means carriage return, `\n` means newline, and `\b` means bell. Any octal value can be specified with `\`*nnn*, where *nnn* is any positive octal number less than 0400. Any hexadecimal value can be specified with `\x`*nn*, where *nn* is any positive hexadecimal number less than or equal to `0xff`.

colon-separated hexadecimal list

A list of hexadecimal octet values, separated by colons, may be specified as a data expression.

`concat` (*data-expr1*, ,,,, *data-exprN*)

The expressions are evaluated, and the results of each evaluation are concatenated in the sequence that the subexpressions are listed. If any subexpression evaluates to null, the result of the concatenation is null.

`reverse` (*numeric-expr1*, *data-expr2*)

The two expressions are evaluated, and then the result of evaluating the data expression is reversed in place, using hunks of the size specified in the

Copyright © 2014, QNX Software Systems Limited

numeric expression. For example, if the numeric expression evaluates to four, and the data expression evaluates to twelve bytes of data, then the reverse expression will evaluate to twelve bytes of data, consisting of the last four bytes of the the input data, followed by the middle four bytes, followed by the first four bytes.

`leased-address`

In any context where the client whose request is being processed has been assigned an IP address, this data expression returns that IP address. In any context where the client whose request is being processed has not been assigned an IP address, if this data expression is found in executable statements executed on that client's behalf, a log message indicating “there is no lease associated with this client” is syslogged to the debug level (this is considered `dhcpd.conf` debugging information).

`binary-to-ascii` (*numeric-expr1*, *numeric-expr2*, *data-expr*, *data-expr2*)

Converts the result of evaluating *data-expr2* into a text string containing one number for each element of the result of evaluating *data-expr2*. Each number is separated from the other by the result of evaluating *data-expr1*. The result of evaluating *numeric-expr1* specifies the base (2 through 16) into which the numbers should be converted. The result of evaluating *numeric-expr2* specifies the width in bits of each number, which may be 8, 16, or 32.

As an example of the preceding three types of expressions, to produce the name of a PTR record for the IP address being assigned to a client, you could write the following expression:

```
concat (binary-to-ascii (10, 8, ".",
                         reverse (1, leased-address)),
                         ".in-addr.arpa.");
```

`encode-int` (*numeric-expr*, *width*)

The *numeric-expr* is evaluated and encoded as a data string of the specified width, in network byte order (most significant byte first). If the numeric expression evaluates to the null value, the result is also null.

`pick-first-value` (*data-expr1* [... *exprn*])

The `pick-first-value` function takes any number of data expressions as its arguments. Each expression is evaluated, starting with the first in the list, until an expression is found that doesn't evaluate to a null value. That expression is returned, and none of the subsequent expressions are evaluated. If all expressions evaluate to a null value, the null value is returned.

`host-decl-name`

Copyright © 2014, QNX Software Systems Limited

The `host-decl-name` function returns the name of the host declaration that matched the client whose request is currently being processed, if any. If no host declaration matched, the result is the null value.

Numeric expressions

Numeric expressions are expressions that evaluate to an integer. In general, the maximum size of such an integer should not be assumed to be representable in fewer than 32 bits, but the precision of such integers may be more than 32 bits.

`extract-int` (*data-expr*, *width*)

The `extract-int` operator extracts an integer value in network byte order from the result of evaluating the specified data expression. Width is the width in bits of the integer to extract. Currently, the only supported widths are 8, 16, and 32. If the evaluation of the data expression doesn't provide sufficient bits to extract an integer of the specified size, the null value is returned.

`lease-time`

The duration of the current lease; that is, the difference between the current time and the time that the lease expires.

number

Any number between zero and the maximum representable size may be specified as a numeric expression.

`client-state`

The current state of the client instance being processed. This is only useful in DHCP client configuration files. Possible values are:

- Booting — the DHCP client is in the INIT state, and doesn't yet have an IP address. The next message transmitted will be a DHCPDISCOVER, which will be broadcast.
- Reboot — the DHCP client is in the INIT-REBOOT state. It has an IP address, but is not yet using it. The next message to be transmitted will be a DHCPREQUEST, which will be broadcast. If no response is heard, the client will bind to its address and move to the BOUND state.
- Select — the DHCP client is in the SELECTING state; it has received at least one DHCPOFFER message, but is waiting to see if it may receive other DHCPOFFER messages from other servers. No messages are sent in the SELECTING state.
- Request — the DHCP client is in the REQUESTING state; it has received at least one DHCPOFFER message, and has chosen which one it will

Copyright © 2014, QNX Software Systems Limited

request. The next message to be sent will be a DHCPREQUEST message, which will be broadcast.

- Bound — the DHCP client is in the BOUND state; it has an IP address. No messages are transmitted in this state.
- Renew — the DHCP client is in the RENEWING state; it has an IP address, and is trying to contact the server to renew it. The next message to be sent will be a DHCPREQUEST message, which will be unicast directly to the server.
- Rebind — the DHCP client is in the REBINDING state; it has an IP address, and is trying to contact any server to renew it. The next message to be sent will be a DHCPREQUEST, which will be broadcast.

Action expressions

`log` (*priority*, *data-expr*)

Logging statements may be used to send information to the standard logging channels. A logging statement includes an optional priority (`fatal`, `error`, `info`, or `debug`), and a data expression.

Logging statements take only a single data expression argument, so if you want to output multiple data values, you will need to use the `concat` operator to concatenate them.

`execute` (*command-path* [, *data-expr1*, ... *data-exprN*]);

The `execute` statement runs an external command. The first argument is a string literal containing the name or path of the command to run. The other arguments, if present, are either string literals or data-expressions which evaluate to text strings, to be passed as command-line arguments to the command. The `execute` statement is synchronous; the program will block until the external command being run has finished.

Lengthy program execution (for example, in an “on commit” in `dhcpd.conf`) may result in bad performance and timeouts. Only external applications with very short execution times are suitable for use.

Passing user-supplied data to an external application might be dangerous. Make sure the external application checks input buffers for validity. Non-printable ASCII characters will be converted into `dhcpd.conf` language octal escapes (`777`), so make sure your external command handles them as such.

Copyright © 2014, QNX Software Systems Limited

It's possible to use the `execute` statement in any context, not only on events. If you put it in a regular scope in the configuration file, you will execute that command every time a scope is evaluated.

Dynamic DNS updates

The DHCP client and server have the ability to dynamically update the Domain Name System. Within the configuration files, you can define how you want the Domain Name System to be updated. These updates are *RFC2136*-compliant, so any DNS server supporting *RFC2136* should be able to accept updates from the DHCP server.

Security

Support for TSIG and DNSSEC is not yet available. When you set your DNS server up to allow updates from the DHCP server or client, you may be exposing it to unauthorized updates. To avoid this, the best you can do right now is to use IP address-based packet filtering to prevent unauthorized hosts from submitting update requests. Obviously, there is currently no way to provide security for client updates - this will require TSIG or DNSSEC, neither of which is yet available in the DHCP distribution.

Dynamic DNS (DDNS) updates are performed by using the `dns-update` expression. The `dns-update` expression is a boolean expression that takes four parameters. If the update succeeds, the result is true. If it fails, the result is false. The four parameters are:

- the resource record type (RR)
- the left hand side of the RR
- the right hand side of the RR
- the ttl that should be applied to the record

The simplest example of the use of the function can be found in the reference section of the *`dhcpd.conf`* (p. 566) file, where events are described. In this example, several statements are being used to make the arguments to the `dns-update`.

In the example, the first argument to the first `dns-update` expression is a data expression that evaluates to the A RR type. The second argument is constructed by concatenating the DHCP host-name option with a text string containing the local domain, in this case `ssd.example.net`. The third argument is constructed by converting the address the client has been assigned from a 32-bit number into an ASCII string with each byte separated by a period (`.`). The fourth argument, the TTL, specifies the amount of time remaining in the lease (note that this isn't really correct, since the DNS server will pass this TTL out whenever a request comes in, even if that is only a few seconds before the lease expires).

If the first `dns-update` statement succeeds, it's followed up with a second update to install a PTR RR. The installation of a PTR record is similar to installing an A RR except that the left hand side of the record is the leased address, reversed, with `.in-`

Copyright © 2014, QNX Software Systems Limited

`addr.arpa` concatenated. The right hand side is the fully qualified domain name of the client to which the address is being leased.

Contributing author

The Internet Systems Consortium DHCP Distribution was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided through Internet Systems Consortium. Information about Internet Systems Consortium can be found at *http://www.isc.org*.

See also:

RFC2132, *RFC2131*

Copyright © 2014, QNX Software Systems Limited

DHCP Options

Dynamic Host Configuration Protocol options

Description:

The Dynamic Host Configuration protocol allows the client to receive options from the DHCP server describing the network configuration and various services that are available on the network. When configuring *dhcpd* (p. 552) or *dhclient* (p. 478), options must often be declared. The syntax for declaring options, and the names and formats of the options that can be declared, are documented here.

Reference: option statements

DHCP option statements always start with the `option` keyword, followed by an option name, followed by option data. The option names and data formats are described below. It isn't necessary to exhaustively specify all DHCP options; only those options which are needed by clients must be specified.

Option data comes in a variety of formats, as defined below:

- The *ip-address* data type can be entered either as an explicit IP address (e.g., 239.254.197.10) or as a domain name (e.g., `haagen.isc.org`). When entering a domain name, be sure that that domain name resolves to a single IP address.
- The *ip6-address* data specifies an IPv6 address, such as ::1 or 3ffe:bbbb:aaaa:aaaa::1.
- The *int32* data type specifies a signed 32-bit integer. The *uint32* data type specifies an unsigned 32-bit integer. The *int16* and *uint16* data types specify signed and unsigned 16-bit integers. The *int8* and *uint8* data types specify signed and unsigned 8-bit integers. Unsigned 8-bit integers are also sometimes referred to as *octets*.
- The *text* data type specifies an NVT ASCII string, which must be enclosed in double quotes. For example, to specify a root-path option, the syntax would be:

```
option root-path "10.0.1.4:/var/tmp/rootfs";
```

- The *domain-name* data type specifies a domain name, which must not be enclosed in double quotes. This data type isn't used for any existing DHCP options. The domain name is stored just as if it were a text option.
- The *domain-list* data type specifies a list of domain names, enclosed in double quotes and separated by commas (`"example.com", "foo.example.com"`).
- The *flag* data type specifies a boolean value. Booleans can be either true or false (or on or off, if that makes more sense to you).

Copyright © 2014, QNX Software Systems Limited

- The *string* data type specifies either an NVT ASCII string enclosed in double quotes, or a series of octets specified in hexadecimal, separated by colons. For example:

  ```
  option dhcp-client-identifier "CLIENT-FOO";
  ```

  or:

  ```
  option dhcp-client-identifier 43:4c:49:45:54:2d:46:4f:4f;
  ```

Setting option values using expressions

Sometimes it's helpful to be able to set the value of a DHCP option based on some value that the client has sent. To do this, you can use expression evaluation; see the *DHCP Conditional Evaluation* (p. 505) entry. To assign the result of an evaluation to an option, define the option as follows:

```
option my-option = expression ;
```

For example:

```
option hostname = binary-to-ascii (16, 8, "-",
                    substring (hardware, 1, 6));
```

Standard DHCPv4 options

The documentation for the various options mentioned below is taken from the latest IETF draft document on DHCP options. Options not listed below may not yet be implemented, but it is possible to use such options by defining them in the configuration file. For more information, see "*Defining new options* (p. 538)," below.

Some of the options documented here are automatically generated by the DHCP server or by clients, and can't be configured by the user. The value of such an option can be used in the configuration file of the receiving DHCP protocol agent (server or client), for example in conditional expressions. However, the value of the option can't be used in the configuration file of the sending agent, because the value is determined only *after* the configuration file has been processed. In this documentation, such options are shown as "not user-configurable".

The standard options are:

`option all-subnets-local` ***flag*****;**

This option specifies whether or not the client may assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected. A value of true indicates that all subnets share the same MTU. A value of false means that the client should assume that some subnets of the directly connected network may have smaller MTUs.

`option arp-cache-timeout` ***uint32*****;**

Copyright © 2014, QNX Software Systems Limited

This option specifies the timeout in seconds for ARP cache entries.

option bcms-controller-address *ip-address* **[**, *ip-address...* **];**

This option configures a list of IPv4 addresses for use as Broadcast and Multicast Controller Servers ("BCMS").

option bcms-controller-names *domain-list*;

This option contains the domain names of local Broadcast and Multicast Controller Servers (BCMS) controllers which the client may use.

option bootfile-name *text*;

This option is used to identify a bootstrap file. If supported by the client, it should have the same effect as the `filename` declaration. BOOTP clients are unlikely to support this option. Some DHCP clients will support it, and others actually require it.

option boot-size *uint16*;

This option specifies the length in 512-octet blocks of the default boot image for the client.

option broadcast-address *ip-address*;

This option specifies the broadcast address in use on the client's subnet. Legal values for broadcast addresses are specified in section 3.2.1.3 of STD 3 (RFC1122).

option cookie-servers *ip-address* **[**, *ip-address...* **];**

The cookie server option specifies a list of RFC 865 cookie servers available to the client. Servers should be listed in order of preference.

option default-ip-ttl *uint8*;

This option specifies the default time-to-live that the client should use on outgoing datagrams.

option default-tcp-ttl *uint8*;

This option specifies the default TTL that the client should use when sending TCP segments. The minimum value is 1.

option default-url *string*;

The format and meaning of this option is not described in any standards document, but is claimed to be in use by Apple Computer. It is not known

Copyright © 2014, QNX Software Systems Limited

what clients may reasonably do if supplied with this option. Use at your own risk.

option dhcp-client-identifier *string*;

This option can be used to specify a DHCP client identifier in a host declaration, so that *dhcpd* (p. 552) can find the host record by matching against the client identifier.

Some DHCP clients, when configured with client identifiers that are ASCII text, will prepend a zero to the ASCII text. So you may need to write:

```
option dhcp-client-identifier "\0foo";
```

rather than:

```
option dhcp-client-identifier "foo";
```

---

option dhcp-lease-time *uint32*;

This option is used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow the client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer. This option is not directly user-configurable in the server; refer to the `max-lease-time` and `default-lease-time` server options in *dhcpd.conf* (p. 566).

option dhcp-max-message-size *uint16*;

This option, when sent by the client, specifies the maximum size of any response that the server sends to the client. When specified on the server, if the client did not send a `dhcp-max-message-size` option, the size specified on the server is used. This works for BOOTP as well as DHCP responses.

option dhcp-message *text*;

This option is used by a DHCP server to provide an error message to a DHCP client in a DHCPNAK message in the event of a failure. A client may use this option in a DHCPDECLINE message to indicate why the client declined the offered parameters.

This option is not user-configurable.

option dhcp-message-type *uint8*;

Copyright © 2014, QNX Software Systems Limited

This option, sent by both client and server, specifies the type of DHCP message contained in the DHCP packet. Possible values (taken directly from *RFC2132*) are:

- 1 — DHCPDISCOVER
- 2 — DHCPOFFER
- 3 — DHCPREQUEST
- 4 — DHCPDECLINE
- 5 — DHCPACK
- 6 — DHCPNAK
- 7 — DHCPRELEASE
- 8 — DHCPINFORM

This option is not user-configurable.

option dhcp-option-overload ***uint8*****;**

This option is used to indicate that the DHCP *sname* or *file* fields (or both) are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options.

If this option is present, the client interprets the specified additional fields after it concludes interpretation of the standard option fields.

Legal values for this option are:

- 1 — the *file* field is used to hold options.
- 2 — the *sname* field is used to hold options.
- 3 — both fields are used to hold options.

This option is not user-configurable.

option dhcp-parameter-request-list ***uint16*** **[,** ***uint16...*** **];**

This option, when sent by the client, specifies which options the client wishes the server to return. Normally, in the ISC DHCP client, this is done using the `request` statement. If this option isn't specified by the client, the DHCP server will normally return every option that is valid in scope and that fits into the reply. When this option is specified on the server, the server returns the specified options. This can be used to force a client to take options that it hasn't requested, and it can also be used to tailor the response of the DHCP server for clients that may need a more limited set of options than those the server would normally return.

option dhcp-rebinding-time ***uint32*****;**

Copyright © 2014, QNX Software Systems Limited

This option specifies the number of seconds from the time a client gets an address until the client transitions to the REBINDING state.

This option is not user-configurable.

option dhcp-renewal-time *uint32*;

This option specifies the number of seconds from the time a client gets an address until the client transitions to the RENEWING state.

This option is not user-configurable.

option dhcp-requested-address *ip-address*;

This option is used by the client in a DHCPDISCOVER to request that a particular IP address be assigned.

This option is not user-configurable.

option dhcp-server-identifier *ip-address*;

This option is used in DHCPOFFER and DHCPREQUEST messages, and may optionally be included in the DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER in order to allow the client to distinguish between lease offers. DHCP clients use the contents of the "server identifier" field as the destination address for any DHCP messages unicast to the DHCP server. DHCP clients also indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message.

The value of this option is the IP address of the server.

This option is not directly user-configurable. See the `server-identifier` server option in *dhcpd.conf* (p. 566).

option domain-name *text*;

This option specifies the domain name that client should use when resolving hostnames via the Domain Name System.

option domain-name-servers *ip-address* **[**, *ip-address*... **]**;

The `domain-name-servers` option specifies a list of Domain Name System (STD 13, RFC 1035) name servers available to the client. Servers should be listed in order of preference.

option domain-search *domain-list*;

The `domain-search` option specifies a "search list" of Domain Names to be used by the client to locate not-fully-qualified domain names. The

Copyright © 2014, QNX Software Systems Limited

difference between this option and historic use of the `domain-name` option for the same ends is that this option is encoded in RFC1035 compressed labels on the wire. For example:

```
option domain-search "example.com", "sales.example.com",
                     "eng.example.com";
```

option extensions-path *text*;

This option specifies the name of a file containing additional options to be interpreted according to the DHCP option format as specified in RFC2132.

option finger-server *ip-address* **[,** *ip-address...* **];**

The Finger server option specifies a list of Finger servers available to the client. Servers should be listed in order of preference.

option font-servers *ip-address* **[,** *ip-address...* **];**

This option specifies a list of X Window System Font servers available to the client. Servers should be listed in order of preference.

option host-name *string*;

This option specifies the name of the client. The name may or may not be qualified with the local domain name (it's preferable to use the `domain-name` option to specify the domain name). See RFC 1035 for character set restrictions. This option is honored by *`dhclient-script`* (p. 487) only if the hostname for the client machine isn't set.

option ieee802-3-encapsulation *flag*;

This option specifies whether or not the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation if the interface is an Ethernet. A value of false indicates that the client should use RFC 894 encapsulation. A value of true means that the client should use RFC 1042 encapsulation.

option ien116-name-servers *ip-address* **[,** *ip-address...* **];**

The `ien116-name-servers` option specifies a list of IEN 116 name servers available to the client. Servers should be listed in order of preference.

option impress-servers *ip-address* **[,** *ip-address...* **];**

The `impress-server` option specifies a list of Imagen Impress servers available to the client. Servers should be listed in order of preference.

option interface-mtu *uint16*;

Copyright © 2014, QNX Software Systems Limited

This option specifies the MTU to use on this interface. The minimum legal value for the MTU is 68.

option ip-forwarding *flag*;

This option specifies whether the client should configure its IP layer for packet forwarding. A value of false means disable IP forwarding, and a value of true means enable IP forwarding.

option irc-server *ip-address* **[,** *ip-address...* **];**

The IRC server option specifies a list of IRC servers available to the client. Servers should be listed in order of preference.

option log-servers *ip-address* **[,** *ip-address...* **];**

The `log-server` option specifies a list of MIT-LCS UDP log servers available to the client. Servers should be listed in order of preference.

option lpr-servers *ip-address* **[,** *ip-address...* **];**

The LPR server option specifies a list of RFC 1179 line printer servers available to the client. Servers should be listed in order of preference.

option mask-supplier *flag*;

This option specifies whether or not the client should respond to subnet mask requests using ICMP. A value of false indicates that the client should not respond. A value of true means that the client should respond.

option max-dgram-reassembly *uint16*;

This option specifies the maximum size datagram that the client should be prepared to reassemble. The minimum legal value is 576.

option merit-dump *text*;

This option specifies the pathname of a file to which the client's core image should be dumped in the event the client crashes. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

option mobile-ip-home-agent *ip-address* **[,** *ip-address...* **];**

This option specifies a list of IP addresses indicating mobile IP home agents available to the client. Agents should be listed in order of preference, although normally there will be only one such agent.

option nds-context *string*;

Copyright © 2014, QNX Software Systems Limited

The `nds-context` option specifies the name of the initial Netware Directory Service for an NDS client.

`option nds-servers` ***ip-address*** **[, *ip-address...*];**

The `nds-servers` option specifies a list of IP addresses of NDS servers.

`option nds-tree-name` ***string*****;**

The `nds-tree-name` option specifies the NDS tree name that the NDS client should use.

`option netbios-dd-server` ***ip-address*** **[, *ip-address...*];**

The NetBIOS datagram distribution server (NBDD) option specifies a list of RFC 1001/1002 NBDD servers listed in order of preference.

`option netbios-name-servers` ***ip-address*** **[, *ip-address...*];**

The NetBIOS name server (NBNS) option specifies a list of RFC 1001/1002 NBNS name servers listed in order of preference. NetBIOS Name Service is currently more commonly referred to as WINS. WINS servers can be specified using the `netbios-name-servers` option.

`option netbios-node-type` ***uint8*****;**

The NetBIOS node type option allows NetBIOS over TCP/IP clients which are configurable to be configured as described in RFC 1001/1002. The value is specified as a single octet which identifies the client type.

Possible node types are:

- 1 — B-node: Broadcast - no WINS
- 2 — P-node: Peer - WINS only
- 4 — M-node: Mixed - broadcast, then WINS
- 8 — H-node: Hybrid - WINS, then broadcast

`option netbios-scope` ***string*****;**

The NetBIOS scope option specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002. See RFC1001, RFC1002, and RFC1035 for character-set restrictions.

`option netinfo-server-address` ***ip-address*** **[, *ip-address...*];**

The `netinfo-server-address` option hasn't been described in any RFC, but has been allocated (and is claimed to be in use) by Apple Computers. It's hard to say if the above is the correct format, or what clients might be expected to do if values were configured. Use at your own risk.

Copyright © 2014, QNX Software Systems Limited

option netinfo-server-tag *text*;

The `netinfo-server-tag` option hasn't been described in any RFC, but has been allocated (and is claimed to be in use) by Apple Computers. It's hard to say if the above is the correct format, or what clients might be expected to do if values were configured. Use at your own risk.

option nis-domain *text*;

This option specifies the name of the client's NIS (Sun Network Information Services) domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.

option nis-servers *ip-address* **[,** *ip-address...* **];**

This option specifies a list of IP addresses indicating NIS servers available to the client. Servers should be listed in order of preference.

option nisplus-domain *text*;

This option specifies the name of the client's NIS+ domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.

option nisplus-servers *ip-address* **[,** *ip-address...* **];**

This option specifies a list of IP addresses indicating NIS+ servers available to the client. Servers should be listed in order of preference.

option nntp-server *ip-address* **[,** *ip-address...* **];**

The NNTP server option specifies a list of NNTP servers available to the client. Servers should be listed in order of preference.

option non-local-source-routing *flag*;

This option specifies whether the client should configure its IP layer to allow forwarding of datagrams with non-local source routes (see Section 3.3.5 of [4] for a discussion of this topic). A value of false means disallow forwarding of such datagrams, and a value of true means allow forwarding.

option ntp-servers *ip-address* **[,** *ip-address...* **];**

This option specifies a list of IP addresses indicating NTP (RFC 1035) servers available to the client. Servers should be listed in order of preference.

option nwip-domain *string*;

The name of the NetWare/IP domain that a NetWare/IP client should use.

Copyright © 2014, QNX Software Systems Limited

option nwip-suboptions *string*;

A sequence of suboptions for NetWare/IP clients; see RFC2242 for details. Normally this option is set by specifying specific NetWare/IP suboptions; for more information, see "*NetWare/IP suboptions* (p. 532)."

option path-mtu-aging-timeout *uint32*;

This option specifies the timeout (in seconds) to use when aging Path MTU values discovered by the mechanism defined in RFC 1191.

option path-mtu-plateau-table *uint16* **[,** *uint16...* **];**

This option specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest. The minimum MTU value can't be smaller than 68.

option perform-mask-discovery *flag*;

This option specifies whether or not the client should perform subnet mask discovery using ICMP. A value of false indicates that the client should not perform mask discovery. A value of true means that the client should perform mask discovery.

option policy-filter *ip-address ip-address* **[,** *ip-address ip-address...***];**

This option specifies policy filters for non-local source routing. The filters consist of a list of IP addresses and masks which specify destination/mask pairs with which to filter incoming source routes.

Any source routed datagram whose next-hop address does not match one of the filters should be discarded by the client. See STD 3 (RFC1122) for further information.

option pop-server *ip-address* **[,** *ip-address...* **];**

The POP3 server option specifies a list of POP3 servers available to the client. Servers should be listed in order of preference.

option resource-location-servers *ip-address* **[,** *ip-address...***];**

This option specifies a list of RFC 887 Resource Location servers available to the client. Servers should be listed in order of preference.

option root-path *text*;

Copyright © 2014, QNX Software Systems Limited

This option specifies the path-name that contains the client's root disk. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

option router-discovery *flag*;

This option specifies whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256. A value of false indicates that the client should not perform router discovery. A value of true means that the client should perform router discovery.

option router-solicitation-address *ip-address*;

This option specifies the address to which the client should transmit router solicitation requests.

option routers *ip-address* **[,** *ip-address...* **];**

A list of IP addresses for routers on the client's subnet. Routers should be listed in order of preference.

option slp-directory-agent *boolean ip-address* **[,** *ip-address...* **];**

This option specifies two things: the IP addresses of one or more Service Location Protocol Directory Agents, and whether the use of these addresses is mandatory. If the initial boolean value is true, the SLP agent should just use the IP addresses given. If the value is false, the SLP agent may additionally do active or passive multicast discovery of SLP agents (see RFC2165 for details).

In this option and the `slp-service-scope option`, the term "SLP Agent" is being used to refer to a Service Location Protocol agent running on a machine that's being configured using the DHCP protocol.

Also be aware that some companies may refer to SLP as NDS. If you have an NDS directory agent whose address you need to configure, the `slp-directory-agent` option should work.

option slp-service-scope *boolean text*;

The Service Location Protocol Service Scope Option specifies two things: a list of service scopes for SLP, and whether the use of this list is mandatory. If the initial boolean value is true, the SLP agent should only use the list of scopes provided in this option; otherwise, it may use its own static configuration in preference to the list provided in this option.

Copyright © 2014, QNX Software Systems Limited

The text string should be a comma-separated list of scopes that the SLP agent should use. It may be omitted, in which case the SLP Agent will use the aggregated list of scopes of all directory agents known to the SLP agent.

option smtp-server *ip-address* **[,** *ip-address...* **];**

The SMTP server option specifies a list of SMTP servers available to the client. Servers should be listed in order of preference.

option static-routes *ip-address ip-address* **[,** *ip-address ip-address...***];**

This option specifies a list of static routes that the client should install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority.

The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination.

The default route (0.0.0.0) is an illegal destination for a static route. To specify the default route, use the `routers` option. Also, please note that this option is not intended for classless IP routing; it does not include a subnet mask. Since classless IP routing is now the most widely deployed routing standard, this option is virtually useless, and is not implemented by any of the popular DHCP clients, for example the Microsoft DHCP client.

option streettalk-directory-assistance-server *ip-address* **[,** *ip-address...***];**

The StreetTalk Directory Assistance (STDA) server option specifies a list of STDA servers available to the client. Servers should be listed in order of preference.

option streettalk-server *ip-address* **[,** *ip-address...* **];**

The StreetTalk server option specifies a list of StreetTalk servers available to the client. Servers should be listed in order of preference.

option subnet-mask *ip-address***;**

The subnet mask option specifies the client's subnet mask as per RFC 950. If no subnet mask option is provided anywhere in scope, as a last resort dhcpd will use the subnet mask from the subnet declaration for the network on which an address is being assigned. However, *any* `subnet-mask` option declaration that's in scope for the address being assigned will override the subnet mask specified in the subnet declaration.

option subnet-selection *string***;**

Sent by the client if an address is required in a subnet other than the one that would normally be selected (based on the relaying address of the

Copyright © 2014, QNX Software Systems Limited

connected subnet the request is obtained from). See RFC 3011. Note that the option number used by this server is 118; this has not always been the defined number, and some clients may use a different value. Use of this option should be regarded as slightly experimental!

This option is not user-configurable in the server.

option swap-server *ip-address*;

The IP address of the client's swap server.

option tcp-keepalive-garbage *flag*;

This option specifies whether or not the client should send TCP keepalive messages with an octet of garbage for compatibility with older implementations. A value of false indicates that a garbage octet should not be sent. A value of true indicates that a garbage octet should be sent.

option tcp-keepalive-interval *uint32*;

This option specifies the interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer. A value of zero indicates that the client should not generate keepalive messages on connections unless specifically requested by an application.

option tftp-server-name *text*;

This option is used to identify a TFTP server and, if supported by the client, should have the same effect as the `server-name` declaration. BOOTP clients are unlikely to support this option. Some DHCP clients will support it, and others actually require it.

option time-offset *int32*;

The time-offset option specifies the offset of the client's subnet in seconds from Coordinated Universal Time (UTC).

option time-servers *ip-address* **[,** *ip-address...* **];**

A list of RFC 868 time servers available to the client. Servers should be listed in order of preference.

option trailer-encapsulation *flag*;

This option specifies whether or not the client should negotiate the use of trailers (RFC 893 [14]) when using the ARP protocol. A value of false indicates that the client should not attempt to use trailers. A value of true means that the client should attempt to use trailers.

Copyright © 2014, QNX Software Systems Limited

option uap-servers *text*;

This option specifies a list of URLs, each pointing to a user authentication service that is capable of processing authentication requests encapsulated in the User Authentication Protocol (UAP). UAP servers can accept either HTTP 1.1 or SSLv3 connections. If the list includes a URL that does not contain a port component, the normal default port is assumed (i.e., port 80 for `http`, and port 443 for `https`). If the list includes a URL that does not contain a path component, the path `/uap` is assumed. If more than one URL is specified in this list, the URLs are separated by spaces.

option user-class *string*;

This option is used by some DHCP clients as a way for users to specify identifying information to the client. This can be used in a similar way to the vendor-class-identifier option, but the value of the option is specified by the user, not the vendor. Most recent DHCP clients have a way in the user interface to specify the value for this identifier, usually as a text string.

option vendor-class-identifier *string*;

This option is used by some DHCP clients to identify the vendor type and possibly the configuration of a DHCP client. The information is a string of bytes whose contents are specific to the vendor and are not specified in a standard. To see what vendor class identifier clients are sending, you can write the following in your DHCP server configuration file:

```
set vendor-string = option vendor-class-identifier;
```

This will result in all entries in the DHCP server lease database file for clients that sent vendor-class-identifier options having a set statement that looks something like this:

```
set vendor-string = "SUNW.Ultra-5_10";
```

The `vendor-class-identifier` option is normally used by the DHCP server to determine the options that are returned in the `vendor-encapsulated-options` option. See "*Vendor encapsulated options* (p. 541)" below for further information.

option vendor-encapsulated-options *string*;

The `vendor-encapsulated-options` option can contain either a single vendor-specific value or one or more vendor-specific suboptions. This option is not normally specified in the DHCP server configuration file; instead, a vendor class is defined for each vendor, vendor class suboptions are defined,

Copyright © 2014, QNX Software Systems Limited

values for those suboptions are defined, and the DHCP server makes up a response on that basis.

Some default behaviors for well-known DHCP client vendors (currently, the Microsoft Windows 2000 DHCP client) are configured automatically, but otherwise this must be configured manually; see "*Vendor encapsulated options* (p. 541)" below for details.

`option vivso` *string*;

The `vivso` option can contain multiple separate options, one for each 32-bit Enterprise ID. Each Enterprise-ID discriminated option then contains additional options whose format is defined by the vendor who holds that ID. This option is usually not configured manually, but rather is configured via intervening option definitions. See "*Vendor encapsulated options* (p. 541)" below for details.

`option www-server` *ip-address* [, *ip-address...*];

The WWW server option specifies a list of WWW servers available to the client. Servers should be listed in order of preference.

`option x-display-manager` *ip-address* [, *ip-address...*];

This option specifies a list of systems that are running the X Window System Display Manager and are available to the client. Addresses should be listed in order of preference.

Relay agent information option

An IETF draft, `draft-ietf-dhc-agent-options-11.txt`, defines a series of encapsulated options that a relay agent can add to a DHCP packet when relaying it to the DHCP server. The server can then make address allocation decisions (or whatever other decisions it wants) based on these options. The server also returns these options in any replies it sends through the relay agent, so that the relay agent can use the information in these options for delivery or accounting purposes.

The current draft defines two options. To reference these options in the `dhcp` server, specify the option space name, "agent", followed by a period, followed by the option name. It is not normally useful to define values for these options in the server, although it is permissible. These options are not supported in the client.

`option agent.circuit-id` *string*;

The `circuit-id` suboption encodes an agent-local identifier of the circuit from which a DHCP client-to-server packet was received. It is intended for use by agents in relaying DHCP responses back to the proper circuit. The format of this option is currently defined to be vendor-dependent, and will

Copyright © 2014, QNX Software Systems Limited

probably remain that way, although the current draft allows for for the possibility of standardizing the format in the future.

option agent.remote-id *string*;

The `remote-id` suboption encodes information about the remote host end of a circuit. Examples of what it might contain include caller ID information, user-name information, remote ATM address, cable modem ID, and similar things. In principal, the meaning is not well specified, and it should generally be assumed to be an opaque object that is administratively guaranteed to be unique to a particular remote end of a circuit.

option agent.DOCSIS-device-class *uint32*;

The `DOCSIS-device-class` suboption is intended to convey information about the host endpoint, hardware, and software, that either the host operating system or the DHCP server may not otherwise be aware of (but the relay is able to distinguish). This is implemented as a 32-bit field (4 octets), each bit representing a flag describing the host in one of these ways. So far, only bit zero (being the least significant bit) is defined in RFC3256. If this bit is set to one, the host is considered a CPE Controlled Cable Modem (CCCM). All other bits are reserved.

option agent.link-selection *ip-address*;

The `link-selection` suboption is provided by relay agents to inform servers what subnet the client is actually attached to. This is useful in those cases where the *giaddr* (where responses must be sent to the relay agent) is not on the same subnet as the client. When this option is present in a packet from a relay agent, the DHCP server will use its contents to find a subnet declared in configuration, and from here take one step further backwards to any shared-network the subnet may be defined within. The client may be given any address within that shared network, as normally appropriate.

The client FQDN suboptions

The Client FQDN option, currently defined in the Internet Draft draft-ietf-dhc-fqdn-option-00.txt is not a standard yet, but is in sufficiently wide use already that we have implemented it. Due to the complexity of the option format, we have implemented it as a suboption space rather than a single option. In general this option should not be configured by the user; instead it should be used as part of an automatic DNS update system.

option fqdn.no-client-update *flag*;

Copyright © 2014, QNX Software Systems Limited

When the client sends this, if it is true, it means the client will not attempt to update its A record. When sent by the server to the client, it means that the client *shouldn't* update its own A record.

option fqdn.server-update *flag*;

When the client sends this to the server, it's requesting that the server update its A record. When sent by the server, it means that the server has updated (or is about to update) the client's A record.

option fqdn.encoded *flag*;

If true, this indicates that the domain name included in the option is encoded in DNS wire format, rather than as plain ASCII text. The client normally sets this to false if it doesn't support DNS wire format in the FQDN option. The server should always send back the same value that the client sent. When this value is set on the configuration side, it controls the format in which the `fqdn.fqdn` suboption is encoded.

option fqdn.rcode1 *flag*;

option fqdn.rcode2 *flag*;

These options specify the result of the updates of the A and PTR records, respectively, and are only sent by the DHCP server to the DHCP client. The values of these fields are those defined in the DNS protocol specification.

option fqdn.fqdn *text*;

Specifies the domain name that the client wishes to use. This can be a fully-qualified domain name, or a single label. If there is no trailing dot character (`.` in the name, it isn't fully qualified, and the server will generally update that name in some locally-defined domain.

option fqdn.hostname --never set--;

This option should never be set, but it can be read back using the `option` and `config-option` operators in an expression, in which case it returns the first label in the `fqdn.fqdn` suboption. For example, if the value of `fqdn.fqdn` is `foo.example.com.`, then `fqdn.hostname` will be `foo`.

option fqdn.domainname --never set--;

This option should never be set, but it can be read back using the `option` and `config-option` operators in an expression, in which case it returns all labels after the first label in the `fqdn.fqdn` suboption; for example, if the value of `fqdn.fqdn` is `foo.example.com.`, then `fqdn.hostname` will be `example.com.`. If this suboption value isn't set, it means that an

Copyright © 2014, QNX Software Systems Limited

unqualified name was sent in the `fqdn` option, or that no `fqdn` option was sent at all.

If you wish to use any of these suboptions, we strongly recommend that you refer to the Client FQDN option draft (or standard, when it becomes a standard). The documentation here is sketchy and incomplete in comparison, and is just intended for reference by people who already understand the Client FQDN option specification.

NetWare/IP suboptions

RFC2242 defines a set of encapsulated options for Novell NetWare/IP clients. To use these options in the dhcp server, specify the option space name, `nwip`, followed by a period, followed by the option name. The following options can be specified:

option nwip.nsq-broadcast *flag*;

If the value is true, the client should use the NetWare Nearest Server Query to locate a NetWare/IP server. The behavior of the Novell client if this suboption is false, or is not present, is not specified.

option nwip.preferred-dss *ip-address* **[,** *ip-address...* **];**

This suboption specifies a list of up to five IP addresses, each of which should be the IP address of a NetWare Domain SAP/RIP server (DSS).

option nwip.nearest-nwip-server *ip-address* **[,** *ip-address...***];**

This suboption specifies a list of up to five IP addresses, each of which should be the IP address of a Nearest NetWare IP server.

option nwip.autoretries *uint8*;

The number of times that a NetWare/IP client should attempt to communicate with a given DSS server at startup.

option nwip.autoretry-secs *uint8*;

The number of seconds that a Netware/IP client should wait between retries when attempting to establish communications with a DSS server at startup.

option nwip.nwip-1-1 *uint8*;

If the value is true, the NetWare/IP client should support NetWare/IP version 1.1 compatibility. This is needed only if the client will be contacting Netware/IP version 1.1 servers.

option nwip.primary-dss *ip-address*;

Copyright © 2014, QNX Software Systems Limited

The IP address of the Primary Domain SAP/RIP Service server (DSS) for this NetWare/IP domain. The NetWare/IP administration utility uses this value as Primary DSS server when configuring a secondary DSS server.

Standard DHCPv6 options

DHCPv6 options differ from DHCPv4 options partially due to using 16-bit code and length tags, but semantically zero-length options are legal in DHCPv6, and multiple options are treated differently. Whereas in DHCPv4 multiple options would be concatenated to form one option, in DHCPv6 they are expected to be individual instantiations. Understandably, many options are not "allowed" to have multiple instances in a packet; normally these are options which are digested by the DHCP protocol software, and not by users or applications.

`option dhcp6.client-id` *`string`*;

This option specifies the client's DHCP Unique Identifier (DUID). DUIDs are similar but different from DHCPv4 client identifiers; there are documented duid types:

- `duid-llt`
- `duid-en`
- `duid-ll`

This value should not be configured, but rather is provided by clients and treated as an opaque identifier key blob by servers.

`option dhcp6.server-id` *`string`*;

This option specifies the server's DUID identifier. You may use this option to configure an opaque binary blob for your server's identifier.

`option dhcp6.ia-na` *`string`*;

The Identity Association for Non-temporary Addresses (`ia-na`) carries assigned addresses that are not temporary addresses for use by the DHCPv6 client. This option is produced by the DHCPv6 server software, and should not be configured.

`option dhcp6.ia-ta` *`string`*;

The Identity Association for Temporary Addresses (`ia-ta`) carries temporary addresses, which may change upon every renewal. There is no support for this in the current DHCPv6 software.

`option dhcp6.ia-addr` *`string`*;

Copyright © 2014, QNX Software Systems Limited

The Identity Association Address option is encapsulated inside `ia-na` or `ia-ta` options in order to represent addresses associated with those IA's. These options are manufactured by the software, so should not be configured.

option dhcp6.oro *uint16* **[,** *uint16*, **...];**

The Option Request Option (ORO) is the DHCPv6 equivalent of the `parameter-request-list`. Clients supply this option to ask servers to reply with options relevant to their needs and use. This option must not be directly configured; use the `request` syntax in *dhclient.conf* (p. 492) instead.

option dhcp6.preference *uint8*;

The `preference` option informs a DHCPv6 client which server is applied during the initial stages of configuration. Once a client is bound to an IA, it will remain bound to that IA until it is no longer valid or has expired. This value may be configured on the server, and is digested by the client software.

option dhcp6.elapsed-time *uint16*;

The `elapsed-time` option is constructed by the DHCPv6 client software, and is potentially consumed by intermediaries. This option should not be configured.

option dhcp6.relay-msg *string*;

The `relay-msg` option is constructed by intervening DHCPv6 relay agent software. This option is entirely used by protocol software, and is not meant for user configuration.

option dhcp6.unicast *ip6-address*;

The `unicast` option is provided by DHCPv6 servers that are willing (or prefer) to receive Renew packets from their clients by exchanging UDP unicasts with them. Normally, DHCPv6 clients will multicast their Renew messages. This may be configured on the server, and should be configured as an address the server is ready to reply to.

option dhcp6.status-code [*string* **] ;**

The `status-code` option is provided by DHCPv6 servers to inform clients of error conditions during protocol communication. This option is manufactured and digested by protocol software, and should not be configured.

option dhcp6.rapid-commit ;

The `rapid-commit` option is a zero-length option that clients use to indicate their desire to enter into rapid-commit with the server. This option is not

Copyright © 2014, QNX Software Systems Limited

supported by the client at this time, and is digested by the server when present, so should not be configured.

option dhcp6.vendor-opts *string*;

The `vendor-opts` option is actually an encapsulated suboption space, in which each Vendor-specific Information Option (VSIO) is identified by a 32-bit Enterprise-ID number. The encapsulated option spaces within these options are defined by the vendors.

For information about using this option, see "*Vendor encapsulated options* (p. 541)," below, in particular the material about the `vsio` option space.

option dhcp6.interface-id *string*;

The `interface-id` option is manufactured by relay agents, and may be used to guide configuration differentiating clients by the interface they are remotely attached to. It does not make sense to configure a value for this option, but it may make sense to inspect its contents.

option dhcp6.reconf-msg *dhcpv6-message*;

The `reconf-msg` option is manufactured by servers, and sent to clients in Reconfigure messages to inform them of what message the client should Reconfigure using. There is no support for DHCPv6 Reconfigure extensions, and this option is documented informationally only.

option dhcp6.reconf-accept ;

The `reconf-accept` option is included by DHCPv6 clients that support the Reconfigure extensions, advertising that they will respond if the server were to ask them to Reconfigure. There is no support for DHCPv6 Reconfigure extensions, and this option is documented informationally only.

option dhcp6.sip-servers-names *domain-list*;

The `sip-servers-names` option allows SIP clients to locate a local SIP server that's to be used for all outbound SIP requests, a so-called "outbound proxy server." If you wish to use manually entered IPv6 addresses instead, see the `sip-servers-addresses` option below.

option dhcp6.sip-servers-addresses *ip6-address* **[,** *ip6-address* **...] ;**

The `sip-servers-addresses` option allows SIP clients to locate a local SIP server that is to be used for all outbound SIP requests, a so-called "outbound proxy server." If you wish to use domain names rather than IPv6 addresses, see the `sip-servers-names` option above.

option dhcp6.name-servers *ip6-address* **[,** *ip6-address* **...] ;**

Copyright © 2014, QNX Software Systems Limited

The `name-servers` option instructs clients about locally available recursive DNS servers. It's easiest to describe this as the `name-server` line in `/etc/resolv.conf`.

`option dhcp6.domain-search` ***domain-list*****;**

The `domain-search` option specifies the client's domain search path to be applied to recursive DNS queries. It's easiest to describe this as the `search` line in `/etc/resolv.conf`.

`option dhcp6.ia-pd` ***string*****;**

The `ia-pd` option is manufactured by clients and servers to create a Prefix Delegation binding, to delegate an IPv6 prefix to the client. There is not yet any support for prefix delegation in this software, and this option is provided informationally only.

`option dhcp6.ia-prefix` ***string*****;**

The `ia-prefix` option is placed inside `ia-pd` options in order to identify the prefix(es) allocated to the client. There is not yet any support for prefix delegation in this software, and this option is provided informationally only.

`option dhcp6.nis-servers` ***ip6-address*** **[,** ***ip6-address*** **...] ;**

The `nis-servers` option identifies, in order, NIS servers available to the client.

`option dhcp6.nisp-servers` ***ip6-address*** **[,** ***ip6-address*** **...] ;**

The `nisp-servers` option identifies, in order, NIS+ servers available to the client.

`option nis-domain-name` ***domain-list*****;**

The `nis-domain-name` option specifies the NIS domain name the client is expected to use, and is related to the `nis-servers` option.

`option nisp-domain-name` ***domain-list*****;**

The `nisp-domain-name` option specifies the NIS+ domain name the client is expected to use, and is related to the `nisp-servers` option.

`option dhcp6.sntp-servers` ***ip6-address*** **[,** ***ip6-address*** **...] ;**

The `sntp-servers` option specifies a list of local SNTP servers available for the client to synchronize their clocks.

`option dhcp6.info-refresh-time` ***uint32*****;**

Copyright © 2014, QNX Software Systems Limited

The `info-refresh-time` option gives DHCPv6 clients using Information-request messages a hint as to how long they should wait between refreshing the information they were given. Note that this option will only be delivered to the client, and be likely to affect the client's behavior, if the client requested the option.

`option dhcp6.bcms-server-d` ***domain-list*****;**

Contains the domain names of local BCMS (Broadcast and Multicast Control Services) controllers that the client may use.

`option dhcp6.bcms-server-a` ***ip6-address*** **[,** ***ip6-address*** **...];**

Contains the IPv6 addresses of local BCMS (Broadcast and Multicast Control Services) controllers that the client may use.

`option dhcp6.remote-id` ***string*****;**

The `remote-id` option is constructed by relay agents, to inform the server of details pertaining to what the relay knows about the client (such as what port it is attached to, and so forth). The contents of this option have some vendor-specific structure (similar to VSIO), but we have chosen to treat this option as an opaque field.

`option dhcp6.subscriber-id`**;**

An opaque field provided by the relay agent, which provides additional information about the subscriber in question. The exact contents of this option depend upon the vendor and/or the operator's configuration of the remote device, and as such is an opaque field.

`option dhcp6.fqdn` ***string*****;**

The `fqdn` option is normally constructed by the client or server, and negotiates the client's Fully Qualified Domain Name, as well as which party is responsible for Dynamic DNS Updates. See "*Client FQDN suboptions* (p. 530)" for full details (the DHCPv4 and DHCPv6 FQDN options use the same `fqdn.` encapsulated space, so are in all ways identical).

`option dhcp6.lq-query` ***string*****;**

Used internally for lease query.

`option dhcp6.client-data` ***string*****;**

Used internally for lease query.

`option dhcp6.clt-time` ***uint32*****;**

Used internally for lease query.

Copyright © 2014, QNX Software Systems Limited

option dhcp6.lq-relay-data *ip6-address string*;

Used internally for lease query.

option dhcp6.lq-client-link *ip6-address* **[,** *ip6-address* **...];**

Used internally for lease query.

Defining new options

The Internet Systems Consortium DHCP client and server provide the capability to define new options. Each DHCP option has a name, a code, and a structure. The name is used by you to refer to the option. The code is a number, used by the DHCP server and client to refer to an option. The structure describes what the contents of an option looks like.

To define a new option, you need to choose a name for it that is not in use for some other option. For example, you can't use `host-name` because the DHCP protocol already defines a `host-name` option, as described above. If an option name doesn't documented here, you can use it, but it's probably a good idea to put some kind of unique string at the beginning so you can be sure that future options don't take your name. For example, you might define an option, `local-host-name`, feeling some confidence that no official DHCP option name will ever start with `local`.

Once you've chosen a name, you must choose a code. All codes between 224 and 254 are reserved as "site-local" DHCP options, so you can pick any one of these for your site (not for your product/application). In *RFC 3942*, site-local space was moved from starting at 128 to starting at 224. In practice, some vendors have interpreted the protocol rather loosely and have used option code values greater than 128 themselves. There's no real way to avoid this problem, and it was thought to be unlikely to cause too much trouble in practice. If you come across a vendor-documented option code in either the new or old site-local spaces, please contact your vendor and inform them about *RFC 3942*.

The structure of an option is simply the format in which the option data appears. The ISC DHCP server currently supports a few simple types, like integers, booleans, strings and IP addresses, and it also supports the ability to define arrays of single types or arrays of fixed sequences of types.

New options are declared as follows:

```
option new-name code new-code = definition ;
```

The values of *new-name* and *new-code* should be the name and the code that you've chosen for the new option. The *definition* should be the definition of the structure of the option.

The following simple option type definitions are supported:

Copyright © 2014, QNX Software Systems Limited

- boolean:

```
option new-name code new-code = boolean ;
```

  An option of type boolean is a flag with a value of either on or off (or true or false). For example:

```
option use-zephyr code 180 = boolean;
option use-zephyr on;
```

- integer:

```
option new-name code new-code = sign integer width ;
```

  The *sign* token should be blank, `unsigned`, or `signed`. The *width* can be 8, 16, or 32, and refers to the number of bits in the integer. So for example, the following two lines show a definition of the `sql-connection-max` option and its use:

```
option sql-connection-max code 192 = unsigned integer 16;
option sql-connection-max 1536;
```

- IP address:

```
option new-name code new-code = ip-address ;
```

  An option whose structure is an IP address can be expressed either as a domain name or as a dotted quad. For example:

```
option sql-server-address code 193 = ip-address;
option sql-server-address sql.example.com;
```

- IP6 address:

```
option new-name code new-code = ip6-address ;
```

  An option whose structure is an IPv6 address must be expressed as a valid IPv6 address. For example:

```
option dhcp6.some-server code 1234 = array of ip6-address;
option dhcp6.some-server 3ffe:bbbb:aaaa:aaaa::1, 3ffe:bbbb:aaaa:aaaa::2;
```

- text:

```
option new-name code new-code = text ;
```

  An option whose type is text will encode an ASCII text string. For example:

```
option sql-default-connection-name code 194 = text;
option sql-default-connection-name "PRODZA";
```

- data string:

```
option new-name code new-code = string ;
```

Copyright © 2014, QNX Software Systems Limited

An option whose type is a data string is essentially just a collection of bytes, and can be specified either as quoted text, like the text type, or as a list of hexadecimal contents separated by colons whose values must be between 0 and FF. For example:

```
option sql-identification-token code 195 = string;
option sql-identification-token 17:23:19:a6:42:ea:99:7c:22;
```

- domain list:

```
option new-name code new-code = domain-list [compressed] ;
```

An option whose type is *domain-list* is an *RFC1035*-formatted (on the wire, "DNS Format") list of domain names, separated by root labels. The optional `compressed` keyword indicates if the option should be compressed relative to the start of the option contents (not the packet contents).

When in doubt, omit the `compressed` keyword. When the software receives an option that's compressed, and the `compressed` keyword is omitted, it will still decompress the option (relative to the option contents field). The keyword only controls whether or not transmitted packets are compressed.

When *domain-list*-formatted options are output as environment variables to *`dhclient-script`* (p. 487), the standard DNS -escape mechanism is used: they're decimal. This is appropriate for direct use in (for example) `/etc/resolv.conf`.

- encapsulation:

```
option new-name code new-code = encapsulate identifier ;
```

An option whose type is `encapsulate` will encapsulate the contents of the option space specified in *identifier*. Examples of encapsulated options in the DHCP protocol as it currently exists include `vendor-encapsulated-options`, `netware-suboptions`, and `relay-agent-information`.

```
option space local;
option local.demo code 1 = text;
option local-encapsulation code 197 = encapsulate local;
option local.demo "demo";
```

- arrays:

Options can contain arrays of any of the above types except for the text and data string types, which aren't currently supported in arrays. An example of an array definition is as follows:

```
option kerberos-servers code 200 = array of ip-address;
option kerberos-servers 10.20.10.1, 10.20.11.1;
```

- records:

Copyright © 2014, QNX Software Systems Limited

Options can also contain data structures consisting of a sequence of data types, which is sometimes called a record type. For example:

```
option contrived-001 code 201 = { boolean, integer 32, text };
option contrived-001 on 1772 "contrivance";
```

It's also possible to have options that are arrays of records, for example:

```
option new-static-routes code 201 = array of {
     ip-address, ip-address, ip-address, integer 8 };
option static-routes
     10.0.0.0 255.255.255.0 net-0-rtr.example.com 1,
     10.0.1.0 255.255.255.0 net-1-rtr.example.com 1,
     10.2.0.0 255.255.224.0 net-2-0-rtr.example.com 3;
```

Vendor-encapsulated options

The DHCP protocol defines the `vendor-encapsulated-options` option, which allows vendors to define their own options that will be sent encapsulated in a standard DHCP option. It also defines the Vendor Identified Vendor Sub Options option (VIVSO), and the DHCPv6 protocol defines the Vendor-specific Information Option (VSIO). The format of all of these options is usually internally a string of options, similarly to other normal DHCP options. The VIVSO and VSIO options differ in that that they contain options that correspond to vendor Enterprise-ID numbers (assigned by IANA), which then contain options according to each Vendor's specifications. You will need to refer to your vendor's documentation in order to form options to their specification.

The value of these options can be set in one of two ways. The first way is to simply specify the data directly, using a text string or a colon-separated list of hexadecimal values. For help in forming these strings, refer to:

- *RFC2132* for the DHCPv4 Vendor-specific Information Option
- *RFC3925* for the DHCPv4 Vendor Identified Vendor Sub Options
- *RFC3315* for the DHCPv6 Vendor-specific Information Option

For example:

```
option vendor-encapsulated-options
    2:4:
     AC:11:41:1:
    3:12:
     73:75:6e:64:68:63:70:2d:73:65:72:76:65:72:31:37:2d:31:
    4:12:
     2f:65:78:70:6f:72:74:2f:72:6f:6f:74:2f:69:38:36:70:63;
option vivso
    00:00:09:bf:0E:
     01:0c:
         48:65:6c:6c:6f:20:77:6f:72:6c:64:21;
option dhcp6.vendor-opts
    00:00:09:bf:
     00:01:00:0c:
         48:65:6c:6c:6f:20:77:6f:72:6c:64:21;
```

The second way of setting the value of these options is to have the DHCP server generate a vendor-specific option buffer. To do this, you must do the following:

1. Define an option space.

Copyright © 2014, QNX Software Systems Limited

2. Define some options in that option space.
3. Provide values for them.
4. Specify that that option space should be used to generate the relevant option.

To define a new option space in which vendor options can be stored, use the `option space` statement:

```
option space name [ [ code width number ] [ length width number  ]  [
hash size number ] ] ;
```

Where the numbers following `code width`, `length width`, and `hash size` respectively identify the number of bytes used to describe option codes, option lengths, and the size in buckets of the hash tables to hold options in this space (most DHCPv4 option spaces use 1-byte codes and lengths, which is the default, whereas most DHCPv6 option spaces use 2-byte codes and lengths).

The code and length widths are used in the DHCP protocol; you must configure these numbers to match the applicable option space you are configuring. They each default to 1. Valid values for code widths are 1, 2, and 4. Valid values for length widths are 0, 1, and 2. Most DHCPv4 option spaces use 1-byte codes and lengths, which is the default, whereas most DHCPv6 option spaces use 2-byte codes and lengths. A zero-byte length produces options similar to the DHCPv6 Vendor-specific Information Option - but not their contents!

The hash size defaults depend on the `code width` selected, and may be 254 or 1009. Valid values range between 1 and 65535. Note that the higher you configure this value, the more memory will be used. It is considered good practice to configure a value that is slightly larger than the estimated number of options you plan to configure within the space. Previous versions of ISC DHCP (up to and including DHCP 3.0.*), this value was fixed at 9973.

The name can then be used in option definitions, as described earlier in this document. For example:

```
option space SUNW code width 1 length width 1 hash size 3;
option SUNW.server-address code 2 = ip-address;
option SUNW.server-name code 3 = text;
option SUNW.root-path code 4 = text;

option space ISC code width 1 length width 1 hash size 3;
option ISC.sample code 1 = text;
option vendor.ISC code 2495 = encapsulate vivso-sample;
option vendor-class.ISC code 2495 = text;

option ISC.sample "configuration text here";
option vendor-class.ISC "vendor class here";

option space docsis code width 2 length width 2 hash size 17;
option docsis.tftp-servers code 32 = array of ip6-address;
option docsis.cablelabs-configuration-file code 33 = text;
option docsis.cablelabs-syslog-servers code 34 = array of ip6-address;
option docsis.device-id code 36 = string;
option docsis.time-servers code 37 = array of ip6-address;
option docsis.time-offset code 38 = signed integer 32;
option vsio.docsis code 4491 = encapsulate docsis;
```


Copyright © 2014, QNX Software Systems Limited

Once you've defined an option space and the format of some options, you can set up scopes that define values for those options, and you can say when to use them. For example, suppose you want to handle two different classes of clients. Using the option space definition shown in the previous example, you can send different option values to different clients based on the `vendor-class-identifier` option that the clients send, as follows:

```
class "vendor-classes" {
  match option vendor-class-identifier;
}

subclass "vendor-classes" "SUNW.Ultra-5_10" {
  vendor-option-space SUNW;
  option SUNW.root-path "/export/root/sparc";
}

subclass "vendor-classes" "SUNW.i86pc" {
  vendor-option-space SUNW;
  option SUNW.root-path "/export/root/i86pc";
}

option SUNW.server-address 172.17.65.1;
option SUNW.server-name "sundhcp-server17-1";

option vivso-sample.sample "Hello world!";

option docsis.tftp-servers ::1;
```

As you can see in the preceding example, regular scoping rules apply, so you can define values that are global in the global scope, and only define values that are specific to a particular class in the local scope. The `vendor-option-space` declaration tells the DHCP server to use options in the SUNW option space to construct the DHCPv4 `vendor-encapsulated-options` option. This is a limitation of that option - the DHCPv4 VIVSO and the DHCPv6 VSIO options can have multiple vendor definitions all at once (even transmitted to the same client), so it is not necessary to configure this.

Contributing author:

The Internet Systems Consortium DHCP Distribution was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided through Internet Systems Consortium. Information about Internet Systems Consortium can be found at *http://www.isc.org*.

See also:

RFC2132, *RFC2131*, *RFC3046*, *RFC3315*

Copyright © 2014, QNX Software Systems Limited

dhcp.client

TCP/IP host configuration utility

You must be `root` to run this utility.

Syntax:

```
dhcp.client [-abdHkmnRru] [-A num] [-D ident] [-h hostname]
            [-I num] [-i interface]
            [-P port] [-p port] [-s host]
            [-T secs] [-t num] &
```

Runs on:

QNX Neutrino

Options:

-A *num*

The `dhcp.client` declines addresses if they fail an ARP probe (see RFC 2131 and RFC 5227). This option sets the number of consecutive ARP probe tests of the assigned address that can fail before `dhcp.client` gives up. The default is 5; a value of 0 turns off ARP probing.

-a

Apply the assigned IP address as an alias instead of overwriting the current configuration.

-b

Request the DHCP server to send its response packets only to the client where appropriate (default is to request the server to broadcast).

-D *ident*

Specify the client identifier. The default identifier is the MAC address.

-d

Write debugging info to the system log, *`syslogd`* (p. 1885).

-H

Copyright © 2014, QNX Software Systems Limited

Don't apply the hostname to the local system, whether supplied by the server or via the -h option. In the latter case, the hostname is still sent to the server via option 12.

The -H option could be useful in cases where the -h option is used to identify the client, and you want to apply a different hostname locally, or you simply want to ignore the server hostname assignment.

-h ***hostname***

Hostname of client (default is supplied by the server if the hostname is available).

-l ***num***

The number of times to poll waiting for interface to be available (default 5). Polling is done every 2 seconds.

You can use this option to make `dhcp.client` wait until the interface it's to use is available. This is useful in a boot environment when you might not know when the driver is running and registered with the TCPIP stack. The exit status is 2 if no interface is found.

-i ***interface***

The name of the interface to configure (e.g. `en0`, `en1`). The default is the first interface found.

-k

Don't set the `CS_DOMAIN` (Domain Name) configuration string if you're using the -m option.

-m

(QNX Neutrino extension) Instead of writing the domain and nameserver data to `/etc/resolv.conf` (the default), write the data to the `_CS_DOMAIN` and `_CS_RESOLVE` memory configuration-defined values (see *confstr()* in the QNX Neutrino *C Library Reference*):

`_CS_DOMAIN`

Domain name.

`_CS_RESOLVE`

Nameserver.

If you specify -m and -n together, the domain is added, but the nameservers aren't. If you specify -m and -k, `_CS_DOMAIN` isn't set.

Copyright © 2014, QNX Software Systems Limited

-n

Don't add the DHCP-supplied nameservers to `/etc/resolv.conf` or `_CS_RESOLVE`. If you also specify -m, the domain is added, but the nameservers aren't. If you don't specify -m, neither the domain nor the nameservers are added.

-P ***port***

DHCP server port (default is `dhcp` port or port 67).

-p ***port***

DHCP client port (default is `dhcpc` port or port 68).

-R

Don't apply the DHCP-supplied default route. If you specify this option, the default route supplied by the DHCP server isn't applied, but it's still supplied to the `dhcp-up` script (see below) in case you wish to apply it later.

-r

Add *.node_number* to the `resolv.conf` file name. Off by default.

The QNX Neutrino RTOS ignores the -r option.

-s ***host***

Accept packets from this server only; ignore responses from other servers.

-T ***sec***

Specify the time, in seconds, to wait for the client to complete (server ACK) its negotiation with the server. This is applied every time the client returns to the initialize (DISCOVER) state. If a timeout occurs, `dhcp.client` terminates with an exit status of 3.

-t ***num***

Attempt to reach the server *num* times before giving up and terminating (the default is forever). Each attempt lasts 1 minute.

You're likely to use this option in combination with the -u option so that `dhcp.client` times out after a specified number of attempts.

-u

Copyright © 2014, QNX Software Systems Limited

Don't move `dhcp.client` to the background until the interface is configured.

This option is useful for spawning `dhcp.client`. The process doesn't move to the background until it has contacted a server and applied a TCP/IP configuration. The exit status is 3 if no server responds.

Description:

The `dhcp.client` obtains the TCP/IP configuration parameters dynamically from a DHCP (Dynamic Host Configuration Protocol) server, then automatically configures your TCP/IP host. You don't have to provide an IP address or any configuration parameters, or run any configuration utilities.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

If `dhcp.client` is terminated, it releases the DHCP address assigned by the server back to the server. If the client is terminated with `SIGPWR`, the address isn't released; the lease will timeout or be continued at client restart (depending on server policies).

You must start *io-pkt** (p. 1007) before starting `dhcp.client`.

The minimum commands to run under QNX Neutrino are:

```
io-pkt-v4 -dne2000 -ptcpip
if_up -p enx
dhcp.client &
if_up enx
```

Or:

```
io-pkt-v4 -dne2000 -ptcpip
dhcp.client -Ix -u
```

If you wish `dhcp.client` to apply the IP address as an alias instead of overwriting the currently assigned IP address, you must pass the -a option. This option is useful if you wish to assign multiple IP addresses to an interface. You must pass the -a option if you wish to use `dhcp.client` and AutoIP (*lsm-autoip.so* (p. 1145)) on the same interface.

By default, `dhcp.client` searches for an unconfigured interface to provide service. If AutoIP is in use, an unconfigured interface will not be available, and the `dhcp.client` will terminate. In order for `dhcp.client` to provide service to an

Copyright © 2014, QNX Software Systems Limited

interface that already has an IP address assigned to it, use the -i option (in combination with -a), and the interface will have both a DHCP and AutoIP IP address assigned to it.

This utility obtains and implements the following information from the DHCP server:

- Broadcast address
- Domain
- Gateway (default route)
- Hostname
- IP address
- Nameserver
- Netmask

/etc/dhcp/dhcp-check

This is an optional script that you can use to see if the DHCP configuration is correct. It should return 0 if the configuration is acceptable, or a non-zero value if it isn't (which results in a `DHCPDECLINE`).

Environment variables, which contain the configuration that was obtained from the server, are passed to this file. When the file is spawned, it doesn't inherit the full environment. For example, the ***PATH*** environment variable isn't available. To determine which variables are available, you can create a script such as this:

```
#!/bin/sh
env > /tmp/config
```

The environment definitions are:

INTERFACE

The interface that was configured (e.g. `INTERFACE=en0`).

IPADDRESS

The client IP address that was obtained from the server (e.g. `IPADDRESS=10.0.0.1`).

NETMASK

The client netmask that was obtained from the server (e.g. `NETMASK=255.0.0.0`).

HOSTNAME

The hostname of the client (e.g. `HOSTNAME=node1`).

BROADCAST

Copyright © 2014, QNX Software Systems Limited

The client broadcast address that was obtained from the server (e.g. `BROADCAST=10.255.255.255`).

GATEWAY

The gateway that the client is to use (e.g. `GATEWAY=10.0.0.2`).

SERVER

The DHCP server's ID (IP address) (e.g. `SERVER=10.0.0.3`).

NAMESERVER1*, *NAMESERVER2

The nameserver that the client is to use (e.g. `NAMESERVER1=10.0.0.4`).

LEASEOBTAINED

The time at which the lease was obtained (e.g. `LEASEOBTAINED=Mon Oct 30 16:46:10 2000`).

LEASEEXPIRES

The time at which the lease expires (e.g. `LEASEEXPIRES=Mon Oct 31 16:46:10 2000`).

RELAYAGENT

A DHCP relay agent forwards packets between `dhcp.client` and the DHCP server if they're on different networks. This is the IP address of the relay agent if it's present.

SERVERNAME

The hostname of the DHCP server.

DOMAIN

The domain supplied by the DHCP server to be added to `/etc/resolv.conf` or CS_DOMAIN (configuration string).

SIADDR

The next server to use in bootstrap. If a FILENAME was supplied, it would be obtained from this server.

The following options are available but not applied by the `dhcp.client` process:

FILENAME

The filename supplied in the server response (e.g. `FILENAME=/bootimg`).

Any other options are defined as environment variables ***OPTION****x*, where *x* is the option number. If the option is known, `dhcp.client` tries to format it as readable

Copyright © 2014, QNX Software Systems Limited

information. If the option isn't known, `dhcp.client` displays each octet as hexadecimal (e.g. `OPTION200= F1 AA 56 42`).

Currently, `dhcp.client` is aware of options 1 to 61.

`/etc/dhcp/dhcp-up`

If this file exists, it is run after a DHCP server has been contacted and the configuration options above have been applied. This file can be a binary program or a script and must be executable (see *chmod* (p. 124)). If the file is a script, the first line must be the command interpreter. For example:

```
#!/bin/sh
```

The environment is the same as for `dhcp-check`.

`/etc/dhcp/dhcp-options`

This file defines the DHCP options that the client wishes to obtain from the DHCP server. You need this file only if you're adding custom DHCP option handling to the `/etc/dhcp/dhcp-up` file. If you add code to the `dhcp-up` script to handle an option, you must also add this option to `/etc/dhcp/dhcp-options`. The options listed in `dhcp-options` file are sent to the server in addition to the options:

- 1 — subnet mask
- 6 — domain name servers
- 12 — hostname
- 15 — domain name
- 3 — gateway
- 28 — broadcast address

which the `dhcp.client` process itself includes.

Here's an example of the `dhcp-options` file:

```
200
150
#This is a comment
90
```

Specify each option on its own line, listing them in order of priority. Comments must be on a separate line; they can be up to 80 characters long.

Based on:

RFC 2131

Files:

The `dhcp.client` utility requires the `libsocket.so` shared library.

Exit status:

Copyright © 2014, QNX Software Systems Limited

0

Success.

1

An error occurred.

2

No interface was found.

3

No server responded.

Errors:

Errors that occur during configuration are reported to the system log.

Copyright © 2014, QNX Software Systems Limited

dhcpd

Dynamic Host Configuration Protocol (DHCP) server

You must be logged in as `root` to start this server.

Syntax:

```
dhcpd  [-p port] [-f] [-d] [-q] [-t|-T] [-4|-6] [-s server]
       [-cf config-file] [-lf lease-file] [-pf pid-file]
       [-tf trace-output-file] [-play trace-playback-file]
       [ if0 [ ...ifN ] ]

dhcpd --version
```

Runs on:

QNX Neutrino

Options:

-4

Run as a DHCPv4 server (the default).

-6

Run as a DHCPv6 server. If you specify this option, the names of the default files include a "6":

IPv4 name	IPv6 name
`dhcpd.conf`	`dhcpd6.conf`
`dhcpd.leases`	`dhcpd6.leases`
`dhcpd.pid`	`dhcpd6.pid`

-cf *config-file*

The configuration file to use. The default is `/etc/dhcpd.conf` for DHCPv4, and `/etc/dhcpd6.conf` for DHCPv6.

-d

Log any messages to standard error, instead of to `syslog`. If you specify this option, `dhcpd` runs in the foreground.

Copyright © 2014, QNX Software Systems Limited

-f

Run as a foreground process, instead of as a daemon.

-lf ***lease-file***

The name of the leases file to use. The default is `/var/db/dhcpd.leases` for DHCPv4, and `/var/db/dhcpd6.leases` for DHCPv6.

-p ***port***

The port to listen on; the default is port 67.

-pf ***pid-file***

The name of the file that `dhcpd` should store its process ID in. The default is `/var/run/dhcpd.pid` for DHCPv4, and `/var/run/dhcpd6.pid` for DHCPv6.

-play ***trace-playback-file***

Specify the name of the file from which to play back transactions that were earlier recorded with the -tf option.

If you use the -play option, you must specify an alternate lease file (using the -lf option), so that the DHCP server won't wipe out your existing lease file with its test data. The DHCP server will refuse to operate in playback mode unless you specify an alternate lease file.

-q

Be quiet; don't display the copyright message on starting up.

-s ***server***

The address to send replies to; the default is the broadcast address (255.255.255.255).

-T

Test the lease database for correct syntax, but don't attempt to perform any network operations.

-t

Test the configuration file for correct syntax, but don't attempt to perform any network operations.

-tf ***trace-output-file***

Copyright © 2014, QNX Software Systems Limited

Specify the name of the file in which to log the startup state and all transactions processed. You can use the -play option to play back the transactions.

--version

Display the version number of dhcpd, and then exit.

***if0* [...if*N*]**

The names of the network interfaces on which to listen for broadcasts. This should be done on systems where dhcpd is unable to identify non-broadcast interfaces, but shouldn't be required on other systems. If you don't specify any interface names on the command line, dhcpd identifies all network interfaces that are up, eliminating non-broadcast interfaces if possible, and listena for DHCP broadcasts on each interface.

Description:

The Internet Systems Consortium DHCP Server, dhcpd, implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). DHCP allows hosts on a TCP/IP network to request and be assigned IP addresses, and also to discover information about the network to which they are attached. BOOTP provides similar functionality, with certain restrictions.

The DHCP protocol allows a host which is unknown to the network administrator to be automatically assigned a new IP address out of a pool of IP addresses for its network. In order for this to work, the network administrator allocates address pools in each subnet and enters them into the dhcpd.conf(5) file.

On startup, dhcpd reads the dhcpd.conf file and stores a list of available addresses on each subnet in memory. When a client requests an address using the DHCP protocol, dhcpd allocates an address for it. Each client is assigned a lease, which expires after an amount of time chosen by the administrator (by default, one day). Before leases expire, the clients to which leases are assigned are expected to renew them in order to continue to use the addresses. Once a lease has expired, the client to which that lease was assigned is no longer permitted to use the leased IP address.

In order to keep track of leases across system reboots and server restarts, dhcpd keeps a list of leases it has assigned in the *dhcpd.leases* (p. 610) file. Before dhcpd grants a lease to a host, it records the lease in this file and makes sure that the contents of the file are flushed to disk. This ensures that even in the event of a system crash, dhcpd will not forget about a lease that it has assigned. On startup, after reading the *dhcpd.conf* (p. 566) file, dhcpd reads the dhcpd.leases file to refresh its memory about what leases have been assigned.

Copyright © 2014, QNX Software Systems Limited

New leases are appended to the end of the `dhcpd.leases` file. In order to prevent the file from becoming arbitrarily large, from time to time `dhcpd` creates a new `dhcpd.leases` file from its in-core lease database. Once this file has been written to disk, the old file is renamed `dhcpd.leases~`, and the new file is renamed `dhcpd.leases`. If the system crashes in the middle of this process, whichever `dhcpd.leases` file remains will contain all the lease information, so there is no need for a special crash recovery process.

BOOTP support is also provided by this server. Unlike DHCP, the BOOTP protocol doesn't provide a protocol for recovering dynamically-assigned addresses once they are no longer needed. It is still possible to dynamically assign addresses to BOOTP clients, but some administrative process for reclaiming addresses is required. By default, leases are granted to BOOTP clients in perpetuity, although the network administrator may set an earlier cutoff date or a shorter lease length for BOOTP leases if that makes sense.

BOOTP clients may also be served in the old standard way, which is to simply provide a declaration in the `dhcpd.conf` file for each BOOTP client, permanently assigning an address to each client.

Whenever changes are made to the `dhcpd.conf` file, `dhcpd` must be restarted. To restart `dhcpd`, send a `SIGTERM` (signal 15) to the process ID contained in `/var/run/dhcpd.pid`, and then reinvoke `dhcpd`. Because the DHCP server database isn't as lightweight as a BOOTP database, `dhcpd` doesn't automatically restart itself when it sees a change to the `dhcpd.conf` file.

Command line

The names of the network interfaces on which `dhcpd` should listen for broadcasts may be specified on the command line. This should be done on systems where `dhcpd` is unable to identify non-broadcast interfaces, but shouldn't be required on other systems. If no interface names are specified on the command line `dhcpd` will identify all network interfaces which are up, eliminating non-broadcast interfaces if possible, and listen for DHCP broadcasts on each interface.

The server either operates as a DHCPv6 server or a DHCP server, but not both at the same time. To run as a DHCPv6 server, use the -6 flag. To run as a DHCP server, use the -4 flag. If neither is used, the default is to run as a DHCPv6 server.

If `dhcpd` should listen on a port other than the standard (port 67), the -p flag may used. It should be followed by the UDP port number on which `dhcpd` should listen. This is mostly useful for debugging purposes.

If `dhcpd` should send replies to an address other than the broadcast address (255.255.255.255), the -s flag may be used. It is followed by either the IP address or the host name to send replies to. This option is supported only in IPv4.

Copyright © 2014, QNX Software Systems Limited

To run dhcpd as a foreground process, rather than allowing it to run as a daemon in the background, the -f flag should be specified. This is useful when running dhcpd under a debugger, or when running it out of inittab on System V systems.

To have dhcpd log to the standard error descriptor, specify the -d flag. This can be useful for debugging, and also at sites where a complete log of all DHCP activity must be kept, but syslogd isn't reliable or otherwise cannot be used. Normally, dhcpd will log all output using the *syslog()* function with the log facility set to LOG_DAEMON. Note that -d implies -f (the daemon will not fork itself into the background).

You can make dhcpd use an alternate configuration file with the -cf flag, an alternate lease file with the -lf flag, or an alternate pid file with the -pf flag. Because of the importance of using the same lease database at all times when running dhcpd in production, these options should be used only for testing lease files or database files in a non-production environment.

When starting dhcpd up from a system startup script (e.g., /etc/rc), it may not be desirable to print out the entire copyright message on startup. To avoid printing this message, the -q flag may be specified.

The DHCP server reads two files on startup: a configuration file, and a lease database. If the -t flag is specified, the server will simply test the configuration file for correct syntax, but will not attempt to perform any network operations. This can be used to test the a new configuration file automatically before installing it.

The -T flag can be used to test the lease database file in a similar way.

The -tf and -play options allow you to specify a file into which the entire startup state of the server and all the transactions it processes are either logged or played back from. This can be useful in submitting bug reports — if you are getting a core dump every so often, you can start the server with the -tf option and then, when the server dumps core, the trace file will contain all the transactions that led up to it dumping core, so that the problem can be easily debugged with -play.

The -play option must be specified with an alternate lease file, using the -lf switch, so that the DHCP server doesn't wipe out your existing lease file with its test data. The DHCP server will refuse to operate in playback mode unless you specify an alternate lease file.

To find the version of dhcpd that will run, use the --version argument. Instead of running, the version will be displayed.

Configuration

The syntax of the dhcpd.conf file is discussed separately. This section should be used as an overview of the configuration process, and the *dhcpd.conf* (p. 566) documentation should be consulted for detailed reference information.

Subnets

Copyright © 2014, QNX Software Systems Limited

The dhcpd server needs to know the subnet numbers and netmasks of all subnets for which it will be providing service. In addition, in order to dynamically allocate addresses, it must be assigned one or more ranges of addresses on each subnet which it can in turn assign to client hosts as they boot. Thus, a very simple configuration providing DHCP support might look like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
}
```

Multiple address ranges may be specified like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  range 239.252.197.113 239.252.197.250;
}
```

If a subnet will only be provided with BOOTP service and no dynamic address assignment, the `range` clause can be left out entirely, but the `subnet` statement must appear.

Lease lengths

DHCP leases can be assigned almost any length from zero seconds to infinity. What lease length makes sense for any given subnet, or for any given installation, will vary depending on the kinds of hosts being served.

For example, in an office environment where systems are added from time to time and removed from time to time, but move relatively infrequently, it might make sense to allow lease times of a month of more. In a final test environment on a manufacturing floor, it may make more sense to assign a maximum lease length of 30 minutes - enough time to go through a simple test procedure on a network appliance before packaging it up for delivery.

It is possible to specify two lease lengths: the default length that will be assigned if a client doesn't ask for any particular lease length, and a maximum lease length. These are specified as clauses to the subnet command:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  default-lease-time 600;
  max-lease-time 7200;
}
```

This particular subnet declaration specifies a default lease time of 600 seconds (ten minutes), and a maximum lease time of 7200 seconds (two hours). Other common values would be 86400 (one day), 604800 (one week) and 2592000 (30 days). Each subnet need not have the same lease--in the case of an office environment and a manufacturing environment served by the same DHCP server, it might make sense to have widely disparate values for default and maximum lease times on each subnet.

BOOTP support

Each BOOTP client must be explicitly declared in the `dhcpd.conf` file. A very basic client declaration will specify the client network interface's hardware address and the

Copyright © 2014, QNX Software Systems Limited

IP address to assign to that client. If the client needs to be able to load a boot file from the server, that file's name must be specified. A simple bootp client declaration might look like this:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
}
```

Options

DHCP (and also BOOTP with Vendor Extensions) provide a mechanism whereby the server can provide the client with information about how to configure its network interface (e.g., subnet mask), and also how the client can access various network services (e.g., DNS, IP routers, and so on).

These options can be specified on a per-subnet basis, and, for BOOTP clients, also on a per-client basis. In the event that a BOOTP client declaration specifies options that are also specified in its subnet declaration, the options specified in the client declaration take precedence. A reasonably complete DHCP configuration might look something like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
  default-lease-time 600 max-lease-time 7200;
  option subnet-mask 255.255.255.0;
  option broadcast-address 239.252.197.255;
  option routers 239.252.197.1;
  option domain-name-servers 239.252.197.2, 239.252.197.3;
  option domain-name "isc.org";
}
```

A BOOTP host on that subnet that needs to be in a different domain and use a different name server might be declared as follows:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
  option domain-name-servers 192.5.5.1;
  option domain-name "vix.com";
}
```

A more complete description of the `dhcpd.conf` file syntax is provided in *dhcpd.conf* (p. 566).

OMAPI

The DHCP server provides the capability to modify some of its configuration while it is running, without stopping it, modifying its database files, and restarting it. This capability is currently provided using OMAPI, an API for manipulating remote objects. OMAPI clients connect to the server using TCP/IP, authenticate, and can then examine the server's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the *dhcpctl*()* API or OMAPI itself. The *dhcpctl_*()* API is a wrapper that

Copyright © 2014, QNX Software Systems Limited

handles some of the housekeeping chores that OMAPI doesn't do automatically. For more information, see *dhcpctl*()* and OMAPI in the QNX Neutrino *C Library Reference*.

OMAPI exports objects, which can then be examined and modified. The DHCP server exports the following objects: lease, host, failover-state and group. Each object has a number of methods that are provided: lookup, create, and destroy. In addition, it is possible to look at attributes that are stored on objects, and in some cases to modify those attributes.

The lease object

Leases can't currently be created or destroyed, but they can be looked up to examine and modify their state.

Leases have the following attributes:

***state* integer lookup, examine**

- 1 — free
- 2 — active
- 3 — expired
- 4 — released
- 5 — abandoned
- 6 — reset
- 7 — backup
- 8 — reserved
- 9 — bootp

***ip-address* data lookup, examine**

The IP address of the lease.

***dhcp-client-identifier* data lookup, examine, update**

The client identifier that the client used when it acquired the lease. Not all clients send client identifiers, so this may be empty.

***client-hostname* data examine, update**

The value the client sent in the host-name option.

***host* handle examine**

The host declaration associated with this lease, if any.

***subnet* handle examine**

The subnet object associated with this lease (the subnet object isn't currently supported).

Copyright © 2014, QNX Software Systems Limited

pool **handle examine**

The pool object associated with this lease (the pool object isn't currently supported).

billing-class **handle examine**

The handle to the class to which this lease is currently billed, if any (the class object isn't currently supported).

hardware-address **data examine, update**

The hardware address (chaddr) field sent by the client when it acquired its lease.

hardware-type **integer examine, update**

The type of the network interface that the client reported when it acquired its lease.

ends **time examine**

The time when the lease's current state ends, as understood by the client.

tstp **time examine**

The time when the lease's current state ends, as understood by the server.

tsfp **time examine**

The adjusted time when the lease's current state ends, as understood by the failover peer (if there is no failover peer, this value is undefined). Generally this value is only adjusted for expired, released, or reset leases while the server is operating in partner-down state, and otherwise is simply the value supplied by the peer.

atsfp **time examine**

The actual tsfp value sent from the peer. This value is forgotten when a lease binding state change is made, to facilitate retransmission logic.

cltt **time examine**

The time of the last transaction with the client on this lease.

The host object

Hosts can be created, destroyed, looked up, examined and modified. If a host declaration is created or deleted using OMAPI, that information will be recorded in the `dhcpd.leases` file. It's permissible to delete host declarations that are declared in the `dhcpd.conf` file.

Copyright © 2014, QNX Software Systems Limited

Hosts have the following attributes:

name **data lookup, examine, modify**

The name of the host declaration. This name must be unique among all host declarations.

group **handle examine, modify**

The named group associated with the host declaration, if there is one.

hardware-address **data lookup, examine, modify**

The link-layer address that will be used to match the client, if any. Only valid if *hardware-type* is also present.

hardware-type **integer lookup, examine, modify**

The type of the network interface that will be used to match the client, if any. Only valid if *hardware-address* is also present.

dhcp-client-identifier **data lookup, examine, modify**

The dhcp-client-identifier option that will be used to match the client, if any.

ip-address **data examine, modify**

A fixed IP address which is reserved for a DHCP client that matches this host declaration. The IP address will only be assigned to the client if it is valid for the network segment to which the client is connected.

statements **data modify**

A list of statements in the format of the `dhcpd.conf` file that will be executed whenever a message from the client is being processed.

known **integer examine, modify**

If nonzero, indicates that a client matching this host declaration will be treated as known in pool permit lists. If zero, the client will not be treated as known.

The group object

Named groups can be created, destroyed, looked up, examined and modified. If a group declaration is created or deleted using OMAPI, that information will be recorded in the `dhcpd.leases` file. It's permissible to delete group declarations that are declared in the `dhcpd.conf` file.

Named groups currently can only be associated with hosts. This allows one set of statements to be efficiently attached to more than one host declaration.

Copyright © 2014, QNX Software Systems Limited

Groups have the following attributes:

***name* data**

The name of the group. All groups that are created using OMAPI must have names, and the names must be unique among all groups.

***statements* data**

A list of statements in the format of the `dhcpd.conf` file that will be executed whenever a message from a client whose host declaration references this group is processed.

The control object

The control object allows you to shut the server down. If the server is doing failover with another peer, it will make a clean transition into the shutdown state and notify its peer, so that the peer can go into partner down, and then record the “recover” state in the lease file so that when the server is restarted, it will automatically resynchronize with its peer.

On shutdown, the server will also attempt to cleanly shut down all OMAPI connections. If these connections don't go down cleanly after five seconds, they are shut down preemptively. It can take as much as 25 seconds from the beginning of the shutdown process to the time that the server actually exits.

To shut the server down, open its control object and set the state attribute to 2.

The failover-state object

The failover-state object is the object that tracks the state of the failover protocol as it's being managed for a given failover peer. The failover object has the following attributes (see *dhcpd.conf* (p. 566) for explanations about what these attributes mean):

***name* data examine**

Indicates the name of the failover peer relationship, as described in the server's `dhcpd.conf` file.

***partner-address* data examine**

Indicates the failover partner's IP address.

***local-address* data examine**

Indicates the IP address that is being used by the DHCP server for this failover pair.

***partner-port* data examine**

Copyright © 2014, QNX Software Systems Limited

Indicates the TCP port on which the failover partner is listening for failover protocol connections.

local-port **data examine**

Indicates the TCP port on which the DHCP server is listening for failover protocol connections for this failover pair.

max-outstanding-updates **integer examine**

Indicates the number of updates that can be outstanding and unacknowledged at any given time, in this failover relationship.

mclt **integer examine**

Indicates the maximum client lead time in this failover relationship.

load-balance-max-secs **integer examine**

Indicates the maximum value for the secs field in a client request before load balancing is bypassed.

load-balance-hba **data examine**

Indicates the load balancing hash bucket array for this failover relationship.

local-state **integer examine, modify**

Indicates the present state of the DHCP server in this failover relationship. Possible values for state are:

- 1 — startup
- 2 — normal
- 3 — communications interrupted
- 4 — partner down
- 5 — potential conflict
- 6 — recover
- 7 — paused
- 8 — shutdown
- 9 — recover done
- 10 — resolution interrupted
- 11 — conflict done
- 254 — recover wait

(Note that some of the above values have changed since DHCP 3.0.x.)

In general, it isn't a good idea to make changes to this state. However, in the case that the failover partner is known to be down, it can be useful to

Copyright © 2014, QNX Software Systems Limited

set the DHCP server's failover state to partner down. At this point the DHCP server will take over service of the failover partner's leases as soon as possible, and will give out normal leases, not leases that are restricted by MCLT. If you do put the DHCP server into the partner-down when the other DHCP server isn't in the partner-down state, but isn't reachable, IP address assignment conflicts are possible, even likely. Once a server has been put into partner-down mode, its failover partner must not be brought back online until communication is possible between the two servers.

partner-state **integer examine**

Indicates the present state of the failover partner.

local-stos **integer examine**

Indicates the time at which the DHCP server entered its present state in this failover relationship.

partner-stos **integer examine**

Indicates the time at which the failover partner entered its present state.

hierarchy **integer examine**

Indicates whether the DHCP server is primary (0) or secondary (1) in this failover relationship.

last-packet-sent **integer examine**

Indicates the time at which the most recent failover packet was sent by this DHCP server to its failover partner.

last-timestamp-received **integer examine**

Indicates the timestamp that was on the failover message most recently received from the failover partner.

skew **integer examine**

Indicates the skew between the failover partner's clock and this DHCP server's clock

max-response-delay **integer examine**

Indicates the time in seconds after which, if no message is received from the failover partner, the partner is assumed to be out of communication.

cur-unacked-updates **integer examine**

Copyright © 2014, QNX Software Systems Limited

Indicates the number of update messages that have been received from the failover partner but not yet processed.

Files:

The dhcpd server depends on the following libraries and binaries:

- `libcrypto.so`
- `libsocket.so`
- `libdhcpctl.so` (built as part of io-pkt/services/dhcp)
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)

It also uses the following configuration files:

/etc/dhcpd6.conf **(p. 566)**

DHCP configuration file for IPv6 (required; you can override it with -cf *config_file* on startup)

/etc/dhcpd.conf **(p. 566)**

DHCP configuration file for IPv4 operation.

/var/db/dhcpd6.leases **(p. 610)**

Leases file for IPv6 (required, database and server ID, needs to be read/write; you can override it with -lf *leases_file* on startup)

/var/db/dhcpd.leases **(p. 610)**

Leases file for IPv4.

`/var/run/dhcpd.pid`

Process ID of dhcpd.

You should generally create an empty leases file.

Contributing author:

dhcpd was originally written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Version 3 of the DHCP server was funded by Nominum, Inc. Information about Internet Systems Consortium is available at *http://www.isc.org/*. Information about Nominum can be found at *http://www.nominum.com/*.

Copyright © 2014, QNX Software Systems Limited

dhcpd.conf, dhcpd6.conf

DHCP configuration files

Name:

- `/etc/dhcpd.conf` for DHCPv4
- `/etc/dhcpd6.conf` for DHCPv6

Description:

The `dhcpd.conf` file contains configuration information for *dhcpd* (p. 552), the Internet Systems Consortium DHCP Server.

The `dhcpd.conf` file is a free-form ASCII text file. It is parsed by the recursive-descent parser built into dhcpd. The file may contain extra tabs and newlines for formatting purposes. Keywords in the file are case-insensitive. Comments may be placed anywhere within the file (except within quotes). Comments begin with the # character and end at the end of the line.

The file essentially consists of a list of statements. Statements fall into two broad categories: parameters and declarations.

Parameter statements either say how to do something (e.g., how long a lease to offer), whether to do something (e.g., should dhcpd provide addresses to unknown clients), or what parameters to provide to the client (e.g., use gateway 220.177.244.7).

Declarations are used to describe the topology of the network, to describe clients on the network, to provide addresses that can be assigned to clients, or to apply a group of parameters to a group of declarations. In any group of parameters and declarations, all parameters must be specified before any declarations which depend on those parameters may be specified.

Declarations about network topology include the `shared-network` and the `subnet` declarations. If clients on a subnet are to be assigned addresses dynamically, a `range` declaration must appear within the `subnet` declaration. For clients with statically assigned addresses, or for installations where only known clients will be served, each such client must have a `host` declaration. If parameters are to be applied to a group of declarations which aren't related strictly on a per-subnet basis, the `group` declaration can be used.

For every subnet which will be served, and for every subnet to which the DHCP server is connected, there must be one `subnet` declaration, which tells `dhcpd` how to recognize that an address is on that subnet. A `subnet` declaration is required for each subnet even if no addresses will be dynamically allocated on that subnet.

Copyright © 2014, QNX Software Systems Limited

Some installations have physical networks on which more than one IP subnet operates. For example, if there is a site-wide requirement that 8-bit subnet masks be used, but a department with a single physical ethernet network expands to the point where it has more than 254 nodes, it may be necessary to run two 8-bit subnets on the same ethernet until such time as a new physical network can be added. In this case, the `subnet` declarations for these two networks must be enclosed in a `shared-network` declaration.

Note that even when the `shared-network` declaration is absent, an empty one is created by the server to contain the `subnet` (and any scoped parameters included in the `subnet`). For practical purposes, this means that "stateless" DHCP clients, which aren't tied to addresses (and therefore subnets) will receive the same configuration as stateful ones.

Some sites may have departments which have clients on more than one subnet, but it may be desirable to offer those clients a uniform set of parameters which are different than what would be offered to clients from other departments on the same subnet. For clients which will be declared explicitly with `host` declarations, these declarations can be enclosed in a `group` declaration along with the parameters which are common to that department. For clients whose addresses will be dynamically assigned, class declarations and conditional declarations may be used to group parameter assignments based on information the client sends.

When a client is to be booted, its boot parameters are determined by consulting that client's `host` declaration (if any), and then consulting any `class` declarations matching the client, followed by the `pool`, `subnet` and `shared-network` declarations for the IP address assigned to the client. Each of these declarations itself appears within a lexical scope, and all declarations at less specific lexical scopes are also consulted for client option declarations. Scopes are never considered twice, and if parameters are declared in more than one scope, the parameter declared in the most specific scope is the one that is used.

When `dhcpd` tries to find a `host` declaration for a client, it first looks for a `host` declaration which has a `fixed-address` declaration that lists an IP address that is valid for the subnet or shared network on which the client is booting. If it doesn't find any such entry, it tries to find an entry which has no `fixed-address` declaration.

Examples

A typical `dhcpd.conf` file will look something like this:

```
global parameters...

subnet 204.254.239.0 netmask 255.255.255.224 {
  subnet-specific parameters...
  range 204.254.239.10 204.254.239.30;
}

subnet 204.254.239.32 netmask 255.255.255.224 {
  subnet-specific parameters...
  range 204.254.239.42 204.254.239.62;
}
```

Copyright © 2014, QNX Software Systems Limited

```
subnet 204.254.239.64 netmask 255.255.255.224 {
  subnet-specific parameters...
  range 204.254.239.74 204.254.239.94;
}

group {
  group-specific parameters...
  host zappo.test.isc.org {
    host-specific parameters...
  }
  host beppo.test.isc.org {
    host-specific parameters...
  }
  host harpo.test.isc.org {
    host-specific parameters...
  }
}
```

Notice that at the beginning of the file, there's a place for global parameters. These might be things such as the organization's domain name, the addresses of the name servers (if they are common to the entire organization), and so on. So, for example:

```
option domain-name "isc.org";
option domain-name-servers ns1.isc.org, ns2.isc.org;
```

As you can see in the above, you can specify host addresses in parameters using their domain names rather than their numeric IP addresses. If a given hostname resolves to more than one IP address (for example, if that host has two ethernet interfaces), then where possible, both addresses are supplied to the client. The most obvious reason for having subnet-specific parameters as shown earlier is that each subnet, of necessity, has its own router. So for the first subnet, for example, there should be something like:

```
option routers 204.254.239.1;
```

Note that the address here is specified numerically. This isn't required; if you have a different domain name for each interface on your router, it's perfectly legitimate to use the domain name for that interface instead of the numeric address. However, in many cases there may be only one domain name for all of a router's IP addresses, and it wouldn't be appropriate to use that name here.

In the sample file, there's also a `group` statement, which provides common parameters for a set of three hosts: `zappo`, `beppo`, and `harpo`. As you can see, these hosts are all in the `test.isc.org` domain, so it might make sense for a group-specific parameter to override the domain name supplied to these hosts:

```
option domain-name "test.isc.org";
```

Also, given the domain they're in, these are probably test machines. If we wanted to test the DHCP leasing mechanism, we might set the lease timeout somewhat shorter than the default:

```
max-lease-time 120;
default-lease-time 120;
```

You may have noticed that while some parameters start with the `option` keyword, some don't. Parameters starting with the `option` keyword correspond to actual DHCP options, while parameters that don't start with the option keyword either control the

Copyright © 2014, QNX Software Systems Limited

behavior of the DHCP server (e.g., how long a lease dhcpd will give out), or specify client parameters that aren't optional in the DHCP protocol (for example, server-name and filename).

In the sample file, each host had host-specific parameters. These could include such things as the hostname option, the name of a file to upload (the filename parameter) and the address of the server from which to upload the file (the next_server parameter). In general, any parameter can appear anywhere that parameters are allowed, and will be applied according to the scope in which the parameter appears.

Imagine that you have a site with a lot of NCD X-Terminals. These terminals come in a variety of models, and you want to specify the boot files for each model. One way to do this would be to have host declarations for each server and group them by model:

```
group {
  filename "Xncd19r";
  next-server ncd-booter;

  host ncd1 { hardware ethernet 0:c0:c3:49:2b:57; }
  host ncd4 { hardware ethernet 0:c0:c3:80:fc:32; }
  host ncd8 { hardware ethernet 0:c0:c3:22:46:81; }
}

group {
  filename "Xncd19c";
  next-server ncd-booter;

  host ncd2 { hardware ethernet 0:c0:c3:88:2d:81; }
  host ncd3 { hardware ethernet 0:c0:c3:00:14:11; }
}

group {
  filename "XncdHMX";
  next-server ncd-booter;

  host ncd1 { hardware ethernet 0:c0:c3:11:90:23; }
  host ncd4 { hardware ethernet 0:c0:c3:91:a7:8; }
  host ncd8 { hardware ethernet 0:c0:c3:cc:a:8f; }
}
```

Address pools

The pool declaration can be used to specify a pool of addresses that will be treated differently than another pool of addresses, even on the same network segment or subnet. For example, you may want to provide a large set of addresses that can be assigned to DHCP clients that are registered to your DHCP server, while providing a smaller set of addresses, possibly with short lease times, that are available for unknown clients. If you have a firewall, you may be able to arrange for addresses from one pool to be allowed access to the Internet, while addresses in another pool aren't, thus encouraging users to register their DHCP clients. To do this, you would set up a pair of pool declarations:

```
subnet 10.0.0.0 netmask 255.255.255.0 {
  option routers 10.0.0.254;

  # Unknown clients get this pool.
  pool {
    option domain-name-servers bogus.example.com;
    max-lease-time 300;
    range 10.0.0.200 10.0.0.253;
    allow unknown-clients;
  }
```

Copyright © 2014, QNX Software Systems Limited

```
  # Known clients get this pool.
  pool {
    option domain-name-servers ns1.example.com, ns2.example.com;
    max-lease-time 28800;
    range 10.0.0.5 10.0.0.199;
    deny unknown-clients;
  }
}
```

It is also possible to set up entirely different subnets for known and unknown clients; address pools exist at the level of shared networks, so address ranges within pool declarations can be on different subnets.

As you can see in the preceding example, pools can have permit lists that control which clients are allowed access to the pool and which aren't. Each entry in a pool's permit list is introduced with the `allow` or `deny` keyword. If a pool has a permit list, then only those clients that match specific entries on the permit list will be eligible to be assigned addresses from the pool. If a pool has a deny list, then only those clients that don't match any entries on the deny list will be eligible. If both permit and deny lists exist for a pool, then only clients that match the permit list and don't match the deny list will be allowed access.

Dynamic address allocation

Address allocation is actually only done when a client is in the INIT state and has sent a DHCPDISCOVER message. If the client thinks it has a valid lease and sends a DHCPREQUEST to initiate or renew that lease, the server has only three choices: it can ignore the DHCPREQUEST, send a DHCPNAK to tell the client it should stop using the address, or send a DHCPACK, telling the client to go ahead and use the address for a while.

If the server finds the address the client is requesting, and that address is available to the client, the server will send a DHCPACK. If the address is no longer available, or the client isn't permitted to have it, the server will send a DHCPNAK. If the server knows nothing about the address, it will remain silent, unless the address is incorrect for the network segment to which the client has been attached and the server is authoritative for that network segment, in which case the server will send a DHCPNAK even though it doesn't know about the address.

There may be a host declaration matching the client's identification. If that host declaration contains a fixed-address declaration that lists an IP address that is valid for the network segment to which the client is connected. In this case, the DHCP server will never do dynamic address allocation. In this case, the client is *required* to take the address specified in the host declaration. If the client sends a DHCPREQUEST for some other address, the server will respond with a DHCPNAK.

When the DHCP server allocates a new address for a client (remember, this happens only if the client has sent a DHCPDISCOVER), it first looks to see if the client already has a valid lease on an IP address, or if there is an old IP address the client had before that hasn't yet been reassigned. In that case, the server will take that address and check it to see if the client is still permitted to use it. If the client is no longer permitted

Copyright © 2014, QNX Software Systems Limited

to use it, the lease is freed if the server thought it was still in use; the fact that the client has sent a DHCPDISCOVER proves to the server that the client is no longer using the lease.

If no existing lease is found, or if the client is forbidden to receive the existing lease, then the server will look in the list of address pools for the network segment to which the client is attached for a lease that isn't in use and that the client is permitted to have. It looks through each pool declaration in sequence (all `range` declarations that appear outside of pool declarations are grouped into a single pool with no permit list). If the permit list for the pool allows the client to be allocated an address from that pool, the pool is examined to see if there is an address available. If so, then the client is tentatively assigned that address. Otherwise, the next pool is tested. If no addresses are found that can be assigned to the client, no response is sent to the client.

If an address is found that the client is permitted to have, and that has never been assigned to any client before, the address is immediately allocated to the client. If the address is available for allocation but has been previously assigned to a different client, the server will keep looking in hopes of finding an address that has never before been assigned to a client.

The DHCP server generates the list of available IP addresses from a hash table. This means that the addresses aren't sorted in any particular order, and so it isn't possible to predict the order in which the DHCP server will allocate IP addresses. Users of previous versions of the ISC DHCP server may have become accustomed to the DHCP server allocating IP addresses in ascending order, but this is no longer possible, and there is no way to configure this behavior with version 3 of the ISC DHCP server.

IP address conflict prevention

The DHCP server checks IP addresses to see if they are in use before allocating them to clients. It does this by sending an ICMP Echo request message to the IP address being allocated. If no ICMP Echo reply is received within a second, the address is assumed to be free. This is done only for leases that have been specified in `range` statements, and only when the lease is thought by the DHCP server to be free — i.e., the DHCP server or its failover peer hasn't listed the lease as in use.

If a response is received to an ICMP Echo request, the DHCP server assumes that there is a configuration error — the IP address is in use by some host on the network that isn't a DHCP client. It marks the address as abandoned, and will not assign it to clients.

If a DHCP client tries to get an IP address, but none is available, but there are abandoned IP addresses, then the DHCP server will attempt to reclaim an abandoned IP address. It marks one IP address as free, and then does the same ICMP Echo request check described previously. If there is no answer to the ICMP Echo request, the address is assigned to the client.

Copyright © 2014, QNX Software Systems Limited

The DHCP server doesn't cycle through abandoned IP addresses if the first IP address it tries to reclaim is free. Rather, when the next DHCPDISCOVER comes in from the client, it will attempt a new allocation using the same method described here, and will typically try a new IP address.

DHCP Failover

This version of the ISC DHCP server supports the DHCP failover protocol as documented in `draft-ietf-dhc-failover-07.txt`. This isn't a final protocol document, and we haven't done interoperability testing with other vendors' implementations of this protocol, so you mustn't assume that this implementation conforms to the standard. If you wish to use the failover protocol, make sure that both failover peers are running the same version of the ISC DHCP server.

The failover protocol allows two DHCP servers (and no more than two) to share a common address pool. Each server will have about half of the available IP addresses in the pool at any given time for allocation. If one server fails, the other server will continue to renew leases out of the pool, and will allocate new addresses out of the roughly half of available addresses that it had when communications with the other server were lost.

It is possible during a prolonged failure to tell the remaining server that the other server is down, in which case the remaining server will (over time) reclaim all the addresses the other server had available for allocation, and begin to reuse them. This is called putting the server into the PARTNER-DOWN state.

You can put the server into the PARTNER-DOWN state either by using the *omshell* (p. 1412) command or by stopping the server, editing the last peer state declaration in the lease file, and restarting the server. If you use this last method, be sure to leave the date and time of the start of the state blank:

```
failover peer name state {
my state partner-down;
peer state state at date;
}
```

When the other server comes back online, it should automatically detect that it has been offline and request a complete update from the server that was running in the PARTNER-DOWN state, and then both servers will resume processing together.

It is possible to get into a dangerous situation: if you put one server into the PARTNER-DOWN state, and then *that* server goes down, and the other server comes back up, the other server will not know that the first server was in the PARTNER-DOWN state, and may issue addresses previously issued by the other server to different clients, resulting in IP address conflicts. Before putting a server into PARTNER-DOWN state, therefore, make sure that the other server will not restart automatically.

The failover protocol defines a primary server role and a secondary server role. There are some differences in how primaries and secondaries act, but most of the differences simply have to do with providing a way for each peer to behave in the opposite way

Copyright © 2014, QNX Software Systems Limited

from the other. So one server must be configured as primary, and the other must be configured as secondary, and it doesn't matter too much which one is which.

Failover startup

When a server starts that hasn't previously communicated with its failover peer, it must establish communications with its failover peer and synchronize with it before it can serve clients. This can happen either because you have just configured your DHCP servers to perform failover for the first time, or because one of your failover servers has failed catastrophically and lost its database.

The initial recovery process is designed to ensure that when one failover peer loses its database and then resynchronizes, any leases that the failed server gave out before it failed will be honored. When the failed server starts up, it notices that it has no saved failover state, and attempts to contact its peer.

When it has established contact, it asks the peer for a complete copy its peer's lease database. The peer then sends its complete database, and sends a message indicating that it is done. The failed server then waits until MCLT has passed, and once MCLT has passed both servers make the transition back into normal operation. This waiting period ensures that any leases the failed server may have given out while out of contact with its partner will have expired.

While the failed server is recovering, its partner remains in the partner-down state, which means that it is serving all clients. The failed server provides no service at all to DHCP clients until it has made the transition into normal operation.

In the case where both servers detect that they have never before communicated with their partner, they both come up in this recovery state and follow the procedure we have just described. In this case, no service will be provided to DHCP clients until MCLT has expired.

Configuring failover

In order to configure failover, you need to write a `peer` declaration that configures the failover protocol, and you need to write peer references in each pool declaration for which you want to do failover. You don't have to do failover for all pools on a given network segment. You mustn't tell one server it's doing failover on a particular address pool and tell the other it isn't. You mustn't have any common address pools on which you aren't doing failover. A pool declaration that utilizes failover would look like this:

```
pool {
     failover peer "foo";
     pool-specific parameters
};
```

The server currently does very little sanity checking, so if you configure it wrong, it will just fail in odd ways. I would recommend therefore that you either do failover or don't do failover, but don't do any mixed pools. Also, use the same master configuration file for both servers, and have a separate file that contains the peer declaration and includes the master file. This will help you to avoid configuration mismatches. As our

Copyright © 2014, QNX Software Systems Limited

implementation evolves, this will become less of a problem. A basic sample `dhcpd.conf` file for a primary server might look like this:

```
failover peer "foo" {
  primary;
  address anthrax.rc.vix.com;
  port 519;
  peer address trantor.rc.vix.com;
  peer port 520;
  max-response-delay 60;
  max-unacked-updates 10;
  mclt 3600;
  split 128;
  load balance max seconds 3;
}

include "/etc/dhcpd.master";
```

The statements in the peer declaration are as follows:

- The `primary` and `secondary` statements:

  ```
  [ primary | secondary ];
  ```

  This determines whether the server is primary or secondary, as described earlier under DHCP FAILOVER.

- The `address` statement:

  ```
  address address;
  ```

  The `address` statement declares the IP address or DNS name on which the server should listen for connections from its failover peer, and also the value to use for the DHCP Failover Protocol server identifier. Because this value is used as an identifier, it may not be omitted.

- The `peer address` statement:

  ```
  peer address address;
  ```

  This statement declares the IP address or DNS name to which the server should connect to reach its failover peer for failover messages.

- The `port` statement:

  ```
  port port_number;
  ```

  The `port` statement declares the TCP port on which the server should listen for connections from its failover peer. This statement may not currently be omitted, because the failover protocol doesn't yet have a reserved TCP port number.

- The `peer port` statement:

  ```
  peer port port_number;
  ```

  The `peer port` statement declares the TCP port to which the server should connect to reach its failover peer for failover messages. This statement may not be

Copyright © 2014, QNX Software Systems Limited

omitted because the failover protocol does not yet have a reserved TCP port number. The port number declared in the `peer port` statement may be the same as the port number declared in the `port` statement.

- The `max-response-delay` statement:

  ```
  max-response-delay seconds;
  ```

  The `max-response-delay` statement tells the DHCP server how many seconds may pass without receiving a message from its failover peer before it assumes that connection has failed. This number should be small enough that a transient network failure that breaks the connection will not result in the servers being out of communication for a long time, but large enough that the server isn't constantly making and breaking connections. This parameter must be specified.

- The `max-unacked-updates` statement:

  ```
  max-unacked-updates count;
  ```

  The `max-unacked-updates` statement tells the remote DHCP server how many BNDUPD messages it can send before it receives a BNDACK from the local system. We don't have enough operational experience to say what a good value for this is, but 10 seems to work. This parameter must be specified.

- The `mclt` statement:

  ```
  mclt seconds;
  ```

  The `mclt` statement defines the Maximum Client Lead Time. It must be specified on the primary, and may not be specified on the secondary. This is the length of time for which a lease may be renewed by either failover peer without contacting the other. The longer you set this, the longer it will take for the running server to recover IP addresses after moving into PARTNER-DOWN state. The shorter you set it, the more load your servers will experience when they aren't communicating. A value of something like 3600 is probably reasonable, but again bear in mind that we have no real operational experience with this.

- The `split` statement:

  ```
  split index;
  ```

  The `split` statement specifies the split between the primary and secondary for the purposes of load balancing. Whenever a client makes a DHCP request, the DHCP server runs a hash on the client identification, resulting in value from 0 to 255. This is used as an index into a 256-bit field. If the bit at that index is set, the primary is responsible. If the bit at that index isn't set, the secondary is responsible. The `split` value determines how many of the leading bits are set to one. So, in practice, higher split values will cause the primary to serve more clients

Copyright © 2014, QNX Software Systems Limited

than the secondary. Lower split values, the converse. Legal values are between 0 and 255, of which the most reasonable is 128.

- The `hba` statement:

  ```
  hba colon-separated-hex-list;
  ```

  The `hba` statement specifies the split between the primary and secondary as a bitmap rather than a cutoff, which theoretically allows for finer-grained control. In practice, there is probably no need for such fine-grained control, however. A sample `hba` statement:

  ```
  hba ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
      00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00;
  ```

  This is equivalent to a `split 128;` statement, and identical. The following two examples are also equivalent to a `split` of 128, but are not identical:

  ```
  hba aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:
      aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa;

  hba 55:55:55:55:55:55:55:55:55:55:55:55:55:55:55:55:
      55:55:55:55:55:55:55:55:55:55:55:55:55:55:55:55;
  ```

  They are equivalent, because half the bits are set to 0, half are set to 1 (0xa and 0x5 are 1010 and 0101 binary respectively) and consequently this would roughly divide the clients equally between the servers. They aren't identical, because the actual peers this would load balance to each server are different for each example.

  You must have only `split` or `hba` defined, never both. For most cases, the fine-grained control that `hba` offers isn't necessary, and `split` should be used.

- The `load balance max seconds` statement:

  ```
  load balance max seconds seconds;
  ```

  This statement allows you to configure a cutoff after which load balancing is disabled. The cutoff is based on the number of seconds since the client sent its first DHCPDISCOVER or DHCPREQUEST message, and only works with clients that correctly implement the `secs` field — fortunately most clients do. We recommend setting this to something like 3 or 5. The effect of this is that if one of the failover peers gets into a state where it is responding to failover messages but not responding to some client requests, the other failover peer will take over its client load automatically as the clients retry.

- The failover pool balance statements:

  ```
  max-lease-misbalance percentage;
  max-lease-ownership percentage;
  min-balance seconds;
  max-balance seconds;
  ```


Copyright © 2014, QNX Software Systems Limited

This version of the DHCP Server evaluates pool balance on a schedule, rather than on demand as leases are allocated. The latter approach proved to be slightly clunky when pool misbalanced reach total saturation; when any server ran out of leases to assign, it also lost its ability to notice it had run dry.

In order to understand pool balance, some elements of its operation first need to be defined. First, there are "free" and "backup" leases. Both of these are referred to as *free state leases*. "Free" and "backup" are the free states for the purpose of this document. The difference is that only the primary may allocate from "free" leases unless under special circumstances, and only the secondary may allocate "backup" leases.

When pool balancing is performed, the only plausible expectation is to provide a 50/50 split of the free state leases between the two servers. This is because no one can predict which server will fail, regardless of the relative load placed upon the two servers, so giving each server half the leases gives both servers the same amount of "failure endurance." Therefore, there is no way to configure any different behavior, outside of some very small windows we will describe shortly.

The first thing calculated on any pool balance run is a value referred to as *lts*, or "Leases To Send". This, simply, is the difference in the count of free and backup leases, divided by two. For the secondary, it is the difference in the backup and free leases, divided by two. The resulting value is signed: if it is positive, the local server is expected to hand out leases to retain a 50/50 balance. If it is negative, the remote server would need to send leases to balance the pool. Once the lts value reaches zero, the pool is perfectly balanced (give or take one lease in the case of an odd number of total free state leases).

The current approach is still something of a hybrid of the old approach, marked by the presence of the `max-lease-misbalance` statement. This parameter configures what used to be a 10% fixed value in previous versions: if *lts* is less than free + backup * `max-lease-misbalance` percent, then the server will skip balancing a given pool (it won't bother moving any leases, even if some leases "should" be moved).

The meaning of this value is also somewhat overloaded, however, in that it also governs the estimation of when to attempt to balance the pool (which may then also be skipped over). The oldest leases in the free and backup states are examined. The time they have resided in their respective queues is used as an estimate to indicate how much time it is probable it would take before the leases at the top of the list would be consumed (and thus, how long it would take to use all leases in that state). This percentage is directly multiplied by this time, and fit into the schedule if it falls within the `min-balance` and `max-balance` configured values. The scheduled pool check time is only moved in a downwards direction; it is never increased. Lastly, if the *lts* is more than double this number in the negative

Copyright © 2014, QNX Software Systems Limited

direction, the local server will “panic” and transmit a Failover protocol POOLREQ message, in the hopes that the remote system will be woken up into action.

Once the *lts* value exceeds the `max-lease-misbalance` percentage of total free state leases as described above, leases are moved to the remote server. This is done in two passes.

In the first pass, only leases whose most recent bound client would have been served by the remote server — according to the Load Balance Algorithm (see above `split` and `hba` configuration statements) — are given away to the peer. This first pass will happily continue to give away leases, decrementing the lts value by one for each, until the *lts* value has reached the negative of the total number of leases multiplied by the `max-lease-ownership` percentage. So it is through this value that you can permit a small misbalance of the lease pools — for the purpose of giving the peer more than a 50/50 share of leases in the hopes that their clients might some day return and be allocated by the peer (operating normally). This process is referred to as *MAC Address Affinity*, but this is somewhat misnamed: it applies equally to DHCP Client Identifier options. Note also that affinity is applied to leases when they enter the state be moved from free to backup if the secondary already has more than its share.

The second pass is entered into only if the first pass fails to reduce the *lts* underneath the total number of free state leases multiplied by the `max-lease-ownership` percentage. In this pass, the oldest leases are given over to the peer without second thought about the Load Balance Algorithm, and this continues until the lts falls under this value. In this way, the local server will also happily keep a small percentage of the leases that would normally load balance to itself.

So, the `max-lease-misbalance` value acts as a behavioral gate. Smaller values will cause more leases to transition states to balance the pools over time, higher values will decrease the amount of change (but may lead to pool starvation if there's a run on leases).

The `max-lease-ownership` value permits a small (percentage) skew in the lease balance of a percentage of the total number of free state leases.

Finally, the `min-balance` and `max-balance` make certain that a scheduled rebalance event happens within a reasonable timeframe (not to be thrown off by, for example, a 7 year old free lease).

Plausible values for the percentages lie between 0 and 100, inclusive, but values over 50 are indistinguishable from one another (once lts exceeds 50% of the free state leases, one server must therefore have 100% of the leases in its respective free state). It is recommended to select a `max-lease-ownership` value that is lower than the value selected for the `max-lease-misbalance` value. The `max-lease-ownership` defaults to 10, and `max-lease-misbalance` defaults to 15.

Copyright © 2014, QNX Software Systems Limited

Plausible values for the `min-balance` and `max-balance` times also range from 0 to (2^32)-1 (or the limit of your local `time_t` value), but default to values 60 and 3600 respectively (to place balance events between 1 minute and 1 hour).

Client classing

Clients can be separated into classes, and treated differently depending on what class they are in. This separation can be done either with a conditional statement, or with a `match` statement within the class declaration. It's possible to specify a limit on the total number of clients within a particular class or subclass that may hold leases at one time, and it's possible to specify automatic subclassing based on the contents of the client packet.

To add clients to classes based on conditional evaluation, you can specify a matching expression in the class statement:

```
class "ras-clients" {
  match if substring (option dhcp-client-identifier, 1, 3) = "RAS";
}
```

Whether you use matching expressions or add statements (or both) to classify clients, you must always write a class declaration for any class that you use. If there will be no match statement and no in-scope statements for a class, the declaration should look like this:

```
class "ras-clients" {
}
```

Subclasses

In addition to classes, it's possible to declare subclasses. A subclass is a class with the same name as a regular class, but with a specific submatch expression which is hashed for quick matching. This is essentially a speed hack; the main difference between five classes with match expressions and one class with five subclasses is that it will be quicker to find the subclasses. Subclasses work as follows:

```
class "allocation-class-1" {
  match pick-first-value (option dhcp-client-identifier, hardware);
}

class "allocation-class-2" {
  match pick-first-value (option dhcp-client-identifier, hardware);
}

subclass "allocation-class-1" 1:8:0:2b:4c:39:ad;
subclass "allocation-class-2" 1:8:0:2b:a9:cc:e3;
subclass "allocation-class-1" 1:0:0:c4:aa:29:44;

subnet 10.0.0.0 netmask 255.255.255.0 {
  pool {
    allow members of "allocation-class-1";
    range 10.0.0.11 10.0.0.50;
  }
  pool {
    allow members of "allocation-class-2";
    range 10.0.0.51 10.0.0.100;
  }
}
```

Copyright © 2014, QNX Software Systems Limited

The data following the class name in the subclass declaration is a constant value to use in matching the `match` expression for the class. When class matching is done, the server will evaluate the match expression and then look the result up in the hash table. If it finds a match, the client is considered a member of both the class and the subclass.

Subclasses can be declared with or without scope. In the above example, the sole purpose of the subclass is to allow some clients access to one address pool, while other clients are given access to the other pool, so these subclasses are declared without scopes. If part of the purpose of the subclass were to define different parameter values for some clients, you might want to declare some subclasses with scopes.

In the above example, if you had a single client that needed some configuration parameters, while most didn't, you might write the following subclass declaration for that client:

```
subclass "allocation-class-2" 1:08:00:2b:a1:11:31 {
  option root-path "samsara:/var/diskless/alphapc";
  filename "/tftpboot/netbsd.alphapc-diskless";
}
```

In this example, we've used subclassing as a way to control address allocation on a per-client basis. However, it's also possible to use subclassing in ways that aren't specific to clients — for example, to use the value of the vendor-class-identifier option to determine what values to send in the vendor-encapsulated-options option. An example of this is shown "*Vendor-encapsulated options* (p. 541)" in the DHCP options entry.

Per-class limits on dynamic address allocation

You may specify a limit to the number of clients in a class that can be assigned leases. The effect of this will be to make it difficult for a new client in a class to get an address. Once a class with such a limit has reached its limit, the only way a new client in that class can get a lease is for an existing client to relinquish its lease, either by letting it expire, or by sending a DHCPRELEASE packet. Classes with lease limits are specified as follows:

```
class "limited-1" {
  lease limit 4;
}
```

This will produce a class in which a maximum of four members may hold a lease at one time.

Spawning classes

It's possible to declare a *spawning class*, a class that automatically produces subclasses based on what the client sends. The reason that spawning classes were created was to make it possible to create lease-limited classes on the fly. The envisioned application is a cable-modem environment where the ISP wishes to provide clients at a particular site with more than one IP address, but doesn't wish to provide such clients with their

Copyright © 2014, QNX Software Systems Limited

own subnet, nor give them an unlimited number of IP addresses from the network segment to which they are connected.

Many cable modem head-end systems can be configured to add a Relay Agent Information option to DHCP packets when relaying them to the DHCP server. These systems typically add a circuit ID or remote ID option that uniquely identifies the customer site. To take advantage of this, you can write a class declaration as follows:

```
class "customer" {
  spawn with option agent.circuit-id;
  lease limit 4;
}
```

Now whenever a request comes in from a customer site, the circuit ID option will be checked against the class's hash table. If a subclass is found that matches the circuit ID, the client will be classified in that subclass and treated accordingly. If no subclass is found matching the circuit ID, a new one will be created and logged in the *dhcpd.leases* (p. 610) file, and the client will be classified in this new class. Once the client has been classified, it will be treated according to the rules of the class, including, in this case, being subject to the per-site limit of four leases.

The use of the subclass spawning mechanism isn't restricted to relay agent options; this particular example is given only because it's a fairly straightforward one.

Combining match, match if, and spawn with

In some cases, it may be useful to use one expression to assign a client to a particular class, and a second expression to put it into a subclass of that class. This can be done by combining the `match if` and `spawn with` statements, or the `match if` and `match` statements. For example:

```
class "jr-cable-modems" {
  match if option dhcp-vendor-identifier = "jrcm";
  spawn with option agent.circuit-id;
  lease limit 4;
}

class "dv-dsl-modems" {
  match if option dhcp-vendor-identifier = "dvdsl";
  spawn with option agent.circuit-id;
  lease limit 16;
}
```

This lets you have two classes that both have the same `spawn with` expression without getting the clients in the two classes confused with each other.

Dynamic DNS updates

The DHCP server has the ability to dynamically update the Domain Name System. Within the configuration files, you can define how you want the Domain Name System to be updated. These updates are RFC 2136 compliant, so any DNS server supporting RFC 2136 should be able to accept updates from the DHCP server.

Two DNS update schemes are currently implemented, and another is planned. The two that are currently available are the ad-hoc DNS update mode and the interim DHCP-DNS interaction draft update mode. If and when the DHCP-DNS interaction

Copyright © 2014, QNX Software Systems Limited

draft and the DHCID draft make it through the IETF standards process, there will be a third mode, which will be the standard DNS update method. The DHCP server must be configured to use one of the two currently-supported methods, or not to do dns updates. This can be done with the `ddns-update-style` configuration parameter.

The ad-hoc Dynamic DNS update scheme

The ad-hoc Dynamic DNS update scheme is now deprecated and doesn't work. In future releases of the ISC DHCP server, this scheme will not likely be available. The interim scheme works, allows for failover, and should now be used. The following description is left here for informational purposes only.

The ad-hoc Dynamic DNS update scheme implemented in this version of the ISC DHCP server is a prototype design, which doesn't have much to do with the standard update method that is being standardized in the IETF DHC working group, but rather implements some very basic, yet useful, update capabilities. This mode doesn't work with the failover protocol because it doesn't account for the possibility of two different DHCP servers updating the same set of DNS records.

For the ad-hoc DNS update method, the client's FQDN is derived in two parts. First, the hostname is determined. Then, the domain name is determined, and appended to the hostname.

The DHCP server determines the client's hostname by first looking for a `ddns-hostname` configuration option, and using that if it's present. If no such option is present, the server looks for a valid hostname in the FQDN option sent by the client. If one is found, it is used; otherwise, if the client sent a host-name option, that is used. Otherwise, if there's a host declaration that applies to the client, the name from that declaration will be used. If none of these applies, the server will not have a hostname for the client, and will not be able to do a DNS update.

The domain name is determined from the `ddns-domainname` configuration option. The default configuration for this option is:

```
option server.ddns-domainname = config-option domain-name;
```

So if this configuration option isn't configured to a different value (overriding the above default), or if a domain-name option hasn't been configured for the client's scope, then the server will not attempt to perform a DNS update.

The client's fully qualified domain name, derived as we have described, is used as the name on which an “A” record will be stored. The A record will contain the IP address that the client was assigned in its lease. If there is already an A record with the same name in the DNS server, no update of either the A or PTR records will occur; this prevents a client from claiming that its hostname is the name of some network server. For example, if you have a fileserver called `fs.sneedville.edu`, and the client claims its hostname is `fs`, no DNS update will be done for that client, and an error message will be logged.

Copyright © 2014, QNX Software Systems Limited

If the A record update succeeds, a PTR record update for the assigned IP address will be done, pointing to the A record. This update is unconditional; it will be done even if another PTR record of the same name exists. Since the IP address has been assigned to the DHCP server, this should be safe.

The current implementation assumes clients only have a single network interface. A client with two network interfaces will see unpredictable behavior. This is considered a bug, and will be fixed in a later release. It may be helpful to enable the `one-lease-per-client` parameter so that roaming clients don't trigger this same behavior.

The DHCP protocol normally involves a four-packet exchange: first the client sends a DHCPDISCOVER message, then the server sends a DHCPOFFER, then the client sends a DHCPREQUEST, then the server sends a DHCPACK. In the current version of the server, the server will do a DNS update after it has received the DHCPREQUEST, and before it has sent the DHCPACK. It sends the DNS update only if it hasn't sent one for the client's address before, in order to minimize the impact on the DHCP server.

When the client's lease expires, the DHCP server (if it is operating at the time, or when next it operates) will remove the client's A and PTR records from the DNS database. If the client releases its lease by sending a DHCPRELEASE message, the server will likewise remove the A and PTR records.

The interim DNS update scheme

The interim DNS update scheme operates mostly according to several drafts that are being considered by the IETF and are expected to become standards, but aren't yet standards, and may not be standardized exactly as currently proposed. These are:

- `draft-ietf-dhc-ddns-resolution-??.txt`
- `draft-ietf-dhc-fqdn-option-??.txt`
- `draft-ietf-dnsext-dhcid-rr-??.txt`

Because our implementation is slightly different than the standard, we will briefly document the operation of this update style here.

The first point to understand about this style of DNS update is that unlike the ad-hoc style, the DHCP server doesn't necessarily always update both the A and the PTR records. The FQDN option includes a flag which, when sent by the client, indicates that the client wishes to update its own A record. In that case, the server can be configured either to honor the client's intentions or ignore them. This is done with the statement `allow client-updates` or the statement `ignore client-updates`. By default, client updates are allowed.

If the server is configured to allow client updates, then if the client sends a fully-qualified domain name in the FQDN option, the server will use that name the client sent in the FQDN option to update the PTR record. For example, let us say that

Copyright © 2014, QNX Software Systems Limited

the client is a visitor from the `radish.org` domain, whose hostname is `jschmoe`. The server is for the `example.org` domain. The DHCP client indicates in the FQDN option that its FQDN is `jschmoe.radish.org.`. It also indicates that it wants to update its own A record. The DHCP server therefore doesn't attempt to set up an A record for the client, but does set up a PTR record for the IP address that it assigns the client, pointing at `jschmoe.radish.org`. Once the DHCP client has an IP address, it can update its own A record, assuming that the `radish.org` DNS server will allow it to do so.

If the server is configured not to allow client updates, or if the client doesn't want to do its own update, the server will simply choose a name for the client from either the `fqdn` option (if present) or the `hostname` option (if present). It will use its own domain name for the client, just as in the ad-hoc update scheme. It will then update both the A and PTR record, using the name that it chose for the client. If the client sends a fully-qualified domain name in the `fqdn` option, the server uses only the leftmost part of the domain name — in the example above, `jschmoe` instead of `jschmoe.radish.org`.

Further, if the `ignore client-updates` directive is used, then the server will in addition send a response in the DHCP packet, using the FQDN Option, that implies to the client that it should perform its own updates if it chooses to do so. With `deny client-updates`, a response is sent which indicates the client may not perform updates.

Also, if the `use-host-decl-names` configuration option is enabled, then the host declaration's `hostname` will be used in place of the `hostname` option, and the same rules will apply as described above.

The other difference between the ad-hoc scheme and the interim scheme is that with the interim scheme, a method is used that allows more than one DHCP server to update the DNS database without accidentally deleting A records that shouldn't be deleted nor failing to add A records that should be added. The scheme works as follows:

- When the DHCP server issues a client a new lease, it creates a text string that is an MD5 hash over the DHCP client's identification (see `draft-ietf-dnsext-dhcid-rr-??.txt` for details). The update adds an A record with the name the server chose and a TXT record containing the hashed identifier string (hashid). If this update succeeds, the server is done.
- If the update fails because the A record already exists, then the DHCP server attempts to add the A record with the prerequisite that there must be a TXT record in the same name as the new A record, and that TXT record's contents must be equal to hashid. If this update succeeds, then the client has its A record and PTR record. If it fails, then the name the client has been assigned (or requested) is in use, and can't be used by the client. At this point the DHCP server gives up trying to do a DNS update for the client until the client chooses a new name.

Copyright © 2014, QNX Software Systems Limited

The interim DNS update scheme is called “interim” for two reasons. First, it doesn't quite follow the drafts. The current versions of the drafts call for a new DHCID RRtype, but this isn't yet available. The interim DNS update scheme uses a TXT record instead. Also, the existing ddns-resolution draft calls for the DHCP server to put a DHCID RR on the PTR record, but the interim update method doesn't do this. It is our position that this isn't useful, and we are working with the author in hopes of removing it from the next version of the draft, or better understanding why it is considered useful.

In addition to these differences, the server also doesn't update very aggressively. Because each DNS update involves a round trip to the DNS server, there's a cost associated with doing updates even if they don't actually modify the DNS database. So the DHCP server tracks whether or not it has updated the record in the past (this information is stored on the lease) and doesn't attempt to update records that it thinks it has already updated.

This can lead to cases where the DHCP server adds a record, and then the record is deleted through some other mechanism, but the server never again updates the DNS because it thinks the data is already there. In this case, the data can be removed from the lease through operator intervention, and once this has been done, the DNS will be updated the next time the client renews.

Dynamic DNS update security

When you set your DNS server up to allow updates from the DHCP server, you may be exposing it to unauthorized updates. To avoid this, you should use *TSIG signatures*, a method of cryptographically signing updates using a shared secret key. As long as you protect the secrecy of this key, your updates should also be secure. Note, however, that the DHCP protocol itself provides no security, and that clients can therefore provide information to the DHCP server which the DHCP server will then use in its updates, with the constraints described previously.

The DNS server must be configured to allow updates for any zone that the DHCP server will be updating. For example, let us say that clients in the `sneedville.edu` domain will be assigned addresses on the 10.10.17.0/24 subnet. In that case, you will need a key declaration for the TSIG key you will be using, and also two zone declarations: one for the zone containing A records that will be updates and one for the zone containing PTR records — for ISC BIND, something like this:

```
key DHCP_UPDATER {
  algorithm HMAC-MD5.SIG-ALG.REG.INT;
  secret pRP5FapFoJ95JEL06sv4PQ==;
};

zone "example.org" {
     type master;
     file "example.org.db";
     allow-update { key DHCP_UPDATER; };
};

zone "17.10.10.in-addr.arpa" {
     type master;
     file "10.10.17.db";
     allow-update { key DHCP_UPDATER; };
};
```

Copyright © 2014, QNX Software Systems Limited

You will also have to configure your DHCP server to do updates to these zones. To do so, you need to add something like this to your `dhcpd.conf` file:

```
key DHCP_UPDATER {
  algorithm HMAC-MD5.SIG-ALG.REG.INT;
  secret pRP5FapFoJ95JEL06sv4PQ==;
};

zone EXAMPLE.ORG. {
  primary 127.0.0.1;
  key DHCP_UPDATER;
}

zone 17.127.10.in-addr.arpa. {
  primary 127.0.0.1;
  key DHCP_UPDATER;
}
```

The `primary` statement specifies the IP address of the name server whose zone information is to be updated.

The zone declarations have to correspond to authority records in your name server; in the above example, there must be an SOA record for `example.org.` and for `17.10.10.in-addr.arpa.`. For example, if there were a subdomain `foo.example.org` with no separate SOA, you couldn't write a zone declaration for `foo.example.org`. Also keep in mind that zone names in your DHCP configuration should end in a period (`.`); this is the preferred syntax. If you don't end your zone name in a period, the DHCP server will figure it out. Also note that in the DHCP configuration, zone names aren't encapsulated in quotes where there are in the DNS configuration.

You should choose your own secret key, of course. The ISC BIND 8 and 9 distributions come with a program for generating secret keys called `dnssec-keygen`. The version that comes with BIND 9 is likely to produce a substantially more random key, so we recommend you use that one even if you aren't using BIND 9 as your DNS server. If you are using BIND 9's `dnssec-keygen`, the above key would be created as follows:

```
dnssec-keygen -a HMAC-MD5 -b 128 -n USER DHCP_UPDATER
```

If you're using the BIND 8 dnskeygen program, the following command will generate a key as seen above:

```
dnskeygen -H 128 -u -c -n DHCP_UPDATER
```

You may wish to enable logging of DNS updates on your DNS server. To do so, you might write a `logging` statement like the following:

```
logging {
     channel update_debug {
          file "/var/log/update-debug.log";
          severity  debug 3;
          print-category yes;
          print-severity yes;
          print-time     yes;
     };
     channel security_info    {
          file "/var/log/named-auth.info";
          severity  info;
          print-category yes;
```


Copyright © 2014, QNX Software Systems Limited

```
        print-severity yes;
        print-time     yes;
    };

    category update { update_debug; };
    category security { security_info; };
};
```

You must create the `/var/log/named-auth.info` and `/var/log/update-debug.log` files before starting the name server. For more information on configuring ISC BIND, consult the documentation that accompanies it.

Reference: events

There are three kinds of events that can happen regarding a lease, and it's possible to declare statements that occur when any of these events happen. These events are the commit event, when the server has made a commitment of a certain lease to a client, the release event, when the client has released the server from its commitment, and the expiry event, when the commitment expires.

To declare a set of statements to execute when an event happens, you must use the `on` statement, followed by the name of the event, followed by a series of statements to execute when the event happens, enclosed in braces. Events are used to implement DNS updates, so you should not define your own event handlers if you are using the built-in DNS update mechanism.

The built-in version of the DNS update mechanism is in a text string towards the top of `server/dhcpd.c`. If you want to use events for things other than DNS updates, and you also want DNS updates, you will have to start out by copying this code into your `dhcpd.conf` file and modifying it.

Reference: declarations

- The `include` statement:

  ```
  include filename;
  ```

  The `include` statement is used to read in a named file, and process the contents of that file as though it were entered in place of the `include` statement.

- The `shared-network` statement:

  ```
  shared-network name {
    [ parameters ]
    [ declarations ]
  }
  ```

  The `shared-network` statement is used to inform the DHCP server that some IP subnets actually share the same physical network. Any subnets in a shared network should be declared within a `shared-network` statement. Parameters specified in the `shared-network` statement will be used when booting clients on those subnets unless parameters provided at the subnet or host level override them. If any subnet in a shared network has addresses available for dynamic

Copyright © 2014, QNX Software Systems Limited

allocation, those addresses are collected into a common pool for that shared network and assigned to clients as needed. There is no way to distinguish on which subnet of a shared network a client should boot. The *name* should be the name of the shared network. This name is used when printing debugging messages, so it should be descriptive for the shared network. The name may have the syntax of a valid domain name (although it will never be used as such), or it may be any arbitrary name, enclosed in quotes.

- The `subnet` statement:

```
subnet subnet-number netmask netmask {
  [ parameters ]
  [ declarations ]
}
```

  The `subnet` statement is used to provide `dhcpd` with enough information to tell whether or not an IP address is on that subnet. It may also be used to provide subnet-specific parameters and to specify what addresses may be dynamically allocated to clients booting on that subnet. Such addresses are specified using the `range` declaration.

  The *subnet-number* should be an IP address or domain name that resolves to the subnet number of the subnet being described. The *netmask* should be an IP address or domain name that resolves to the subnet mask of the subnet being described. The subnet number and the netmask are sufficient to determine whether any given IP address is on the specified subnet.

  Although a netmask must be given with every subnet declaration, it's recommended that if there is any variance in subnet masks at a site, a subnet-mask option statement be used in each subnet declaration to set the desired subnet mask, since any subnet-mask option statement will override the subnet mask declared in the subnet statement.

- The `subnet6` statement:

```
subnet6 subnet6-number {
  [ parameters ]
  [ declarations ]
}
```

  The `subnet6` statement is used to provide `dhcpd` with enough information to tell whether or not an IPv6 address is on that subnet6. It may also be used to provide subnet-specific parameters and to specify what addresses may be dynamically allocated to clients booting on that subnet.

  The *subnet6-number* should be an IPv6 network identifier, specified as ip6-address/bits.

- The `range` statement:

```
range [ dynamic-bootp ] low-address [ high-address];
```


Copyright © 2014, QNX Software Systems Limited

For any subnet on which addresses will be assigned dynamically, there must be at least one `range` statement. The range statement gives the lowest and highest IP addresses in a range. All IP addresses in the range should be in the subnet in which the `range` statement is declared. The `dynamic-bootp` flag may be specified if addresses in the specified range may be dynamically assigned to BOOTP clients as well as DHCP clients. When you're specifying a single address, you can omit *high-address*.

- The `range6` statement:

```
range6 low-address high-address;
range6 subnet6-number;
range6 subnet6-number temporary;
range6 address temporary;
```

For any IPv6 subnet6 on which addresses will be assigned dynamically, there must be at least one `range6` statement. The `range6` statement can either be the lowest and highest IPv6 addresses in a `range6`, or use CIDR notation, specified as ip6-address/bits. All IP addresses in the `range6` should be in the subnet6 in which the `range6` statement is declared.

The `temporary` variant makes the prefix (by default on 64 bits) available for temporary (RFC 4941) addresses. A new address per prefix in the shared network is computed at each request with an IA_TA option. Release and Confirm ignores temporary addresses.

Any IPv6 addresses given to hosts with `fixed-address6` are excluded from the `range6`, as are IPv6 addresses on the server itself.

- The `prefix6` statement:

```
prefix6 low-address high-address / bits;
```

The `prefix6` is the `range6` equivalent for Prefix Delegation (RFC 3633). Prefixes of *bits* length are assigned between *low-address* and *high-address*.

Any IPv6 prefixes given to static entries (hosts) with `fixed-prefix6` are excluded from the `prefix6`.

This statement is currently global, but it should have a shared-network scope.

- The `host` statement:

```
host hostname {
  [ parameters ]
  [ declarations ]
}
```

The `host` declaration provides a scope in which to provide configuration information about a specific client, and also provides a way to assign to a client a fixed address. The host declaration provides a way for the DHCP server to identify a DHCP or BOOTP client, and also a way to assign the client a static IP address.

Copyright © 2014, QNX Software Systems Limited

If it is desirable to be able to boot a DHCP or BOOTP client on more than one subnet with fixed addresses, more than one address may be specified in the `fixed-address` declaration, or more than one `host` statement may be specified matching the same client.

If client-specific boot parameters must change based on the network to which the client is attached, then multiple `host` declarations should be used. The `host` declarations will only match a client if one of their `fixed-address` statements is viable on the subnet (or shared network) where the client is attached. Conversely, for a `host` declaration to match a client being allocated a dynamic address, it must not have any `fixed-address` statements. You may therefore need a mixture of `host` declarations for any given client, some with `fixed-address` statements, others without.

The `hostname` should be a name identifying the host. If a `hostname` option isn't specified for the host, `hostname` is used.

The `host` declarations are matched to actual DHCP or BOOTP clients by matching the `dhcp-client-identifier` option specified in the `host` declaration to the one supplied by the client, or, if the `host` declaration or the client doesn't provide a dhcp-client-identifier option, by matching the `hardware` parameter in the `host` declaration to the network hardware address supplied by the client. BOOTP clients don't normally provide a `dhcp-client-identifier`, so the hardware address must be used for all clients that may boot using the BOOTP protocol.

DHCPv6 servers can use the `host-identifier option` parameter in the `host` declaration, and specify any option with a fixed value to identify hosts.

Only the `dhcp-client-identifier` option and the hardware address can be used to match a host declaration, or the `host-identifier option` parameter for DHCPv6 servers. For example, it isn't possible to match a host declaration to a `host-name` option. This is because the host-name option can't be guaranteed to be unique for any given client, whereas both the hardware address and `dhcp-client-identifier` option are at least theoretically guaranteed to be unique to a given client.

- The `group` statement:

```
group {
  [ parameters ]
  [ declarations ]
}
```

The `group` statement is used simply to apply one or more parameters to a group of declarations. It can be used to group hosts, shared networks, subnets, or even other groups.

Reference: allow and deny

Copyright © 2014, QNX Software Systems Limited

The `allow` and `deny` statements can be used to control the response of the DHCP server to various sorts of requests. The `allow` and `deny` keywords actually have different meanings depending on the context. In a pool context, these keywords can be used to set up access lists for address allocation pools. In other contexts, the keywords simply control general server behavior with respect to clients based on scope. In a non-pool context, the `ignore` keyword can be used in place of the `deny` keyword to prevent logging of denied requests.

Allow, deny, and ignore in scope

The following usages of `allow` and `deny` will work in any scope, although it isn't recommended that they be used in pool declarations.

- The `unknown-clients` keyword:

  ```
  allow unknown-clients;
  deny unknown-clients;
  ignore unknown-clients;
  ```

  The `unknown-clients` flag is used to tell `dhcpd` whether or not to dynamically assign addresses to unknown clients. Dynamic address assignment to unknown clients is *allowed* by default. An unknown client is simply a client that has no host declaration.

  The use of this option is now *deprecated*. If you're trying to restrict access on your network to known clients, you should use `deny unknown-clients;` inside of your address pool, as described in "*Allow and deny within pool declarations* (p. 593)," below.

- The `bootp` keyword:

  ```
  allow bootp;
  deny bootp;
  ignore bootp;
  ```

  The `bootp` flag is used to tell `dhcpd` whether or not to respond to bootp queries. Bootp queries are *allowed* by default.

  This option doesn't satisfy the requirement of failover peers for denying dynamic bootp clients. The `deny dynamic bootp clients;` option should be used instead. See "*Allow and deny within pool declarations* (p. 593)" for more details.

- The `booting` keyword:

  ```
  allow booting;
  deny booting;
  ignore booting;
  ```

  The `booting` flag is used to tell `dhcpd` whether or not to respond to queries from a particular client. This keyword has meaning only when it appears in a host declaration. By default, booting is allowed, but if it's disabled for a particular client, then that client will not be able to get an address from the DHCP server.

Copyright © 2014, QNX Software Systems Limited

- The `duplicates` keyword:

  ```
  allow duplicates;
  deny duplicates;
  ```

  Host declarations can match client messages based on the DHCP Client Identifier option or based on the client's network hardware type and MAC address. If the MAC address is used, the host declaration will match any client with that MAC address — even clients with different client identifiers. This doesn't normally happen, but is possible when one computer has more than one operating system installed on it — for example, Microsoft Windows and NetBSD or Linux.

  The `duplicates` flag tells the DHCP server that if a request is received from a client that matches the MAC address of a host declaration, any other leases matching that MAC address should be discarded by the server, even if the UID isn't the same. This is a violation of the DHCP protocol, but can prevent clients whose client identifiers change regularly from holding many leases at the same time. By default, duplicates are allowed.

- The `declines` keyword:

  ```
  allow declines;
  deny declines;
  ignore declines;
  ```

  The DHCPDECLINE message is used by DHCP clients to indicate that the lease the server has offered isn't valid. When the server receives a DHCPDECLINE for a particular address, it normally abandons that address, assuming that some unauthorized system is using it. Unfortunately, a malicious or buggy client can, using DHCPDECLINE messages, completely exhaust the DHCP server's allocation pool. The server will reclaim these leases, but while the client is running through the pool, it may cause serious thrashing in the DNS, and it will also cause the DHCP server to forget old DHCP client address allocations.

  The `declines` flag tells the DHCP server whether or not to honor DHCPDECLINE messages. If it's set to `deny` or `ignore` in a particular scope, the DHCP server will not respond to DHCPDECLINE messages.

- The `client-updates` keyword:

  ```
  allow client-updates;
  deny client-updates;
  ```

  The `client-updates` flag tells the DHCP server whether or not to honor the client's intention to do its own update of its A record. This is relevant only when doing interim DNS updates. See “*The interim DNS update scheme* (p. 583)” for details.

- The `leasequery` keyword:

  ```
  allow leasequery;
  ```


Copyright © 2014, QNX Software Systems Limited

```
deny leasequery;
```

The `leasequery` flag tells the DHCP server whether or not to answer DHCPLEASEQUERY packets. The answer to a DHCPLEASEQUERY packet includes information about a specific lease, such as when it was issued and when it will expire. By default, the server will not respond to these packets.

Allow and deny within pool declarations

The uses of the `allow` and `deny` keywords shown in the previous section work pretty much the same way whether the client is sending a DHCPDISCOVER or a DHCPREQUEST message: an address will be allocated to the client (either the old address it's requesting, or a new address) and then that address will be tested to see if it's okay to let the client have it. If the client requested it, and it's not okay, the server will send a DHCPNAK message. Otherwise, the server will simply not respond to the client. If it is okay to give the address to the client, the server will send a DHCPACK message.

The primary motivation behind pool declarations is to have address allocation pools whose allocation policies are different. A client may be denied access to one pool, but allowed access to another pool on the same network segment. In order for this to work, access control has to be done during address allocation, not after address allocation is done.

When a DHCPREQUEST message is processed, address allocation simply consists of looking up the address the client is requesting and seeing if it's still available for the client. If it is, then the DHCP server checks both the address pool permit lists and the relevant in-scope `allow` and `deny` statements to see if it's okay to give the lease to the client. In the case of a DHCPDISCOVER message, the allocation process is done as described previously in the ADDRESS ALLOCATION section.

When you're declaring permit lists for address allocation pools, the following syntaxes are recognized following the allow or deny keywords:

`known-clients;`

If specified, this statement either allows or prevents allocation from this pool to any client that has a host declaration (i.e., is known). A client is known if it has a host declaration in *any* scope, not just the current scope.

`unknown-clients;`

If specified, this statement either allows or prevents allocation from this pool to any client that has no host declaration (i.e., isn't known).

`members of "`*class*`";`

If specified, this statement either allows or prevents allocation from this pool to any client that is a member of the named class.

Copyright © 2014, QNX Software Systems Limited

`dynamic bootp clients;`

If specified, this statement either allows or prevents allocation from this pool to any bootp client.

`authenticated clients;`

If specified, this statement either allows or prevents allocation from this pool to any client that has been authenticated using the DHCP authentication protocol. This isn't yet supported.

`unauthenticated clients;`

If specified, this statement either allows or prevents allocation from this pool to any client that has not been authenticated using the DHCP authentication protocol. This isn't yet supported.

`all clients;`

If specified, this statement either allows or prevents allocation from this pool to all clients. This can be used when you want to write a pool declaration for some reason, but hold it in reserve, or when you want to renumber your network quickly, and thus want the server to force all clients that have been allocated addresses from this pool to obtain new addresses immediately when they next renew.

`after` ***time*****;**

If specified, this statement either allows or prevents allocation from this pool after a given date. This can be used when you want to move clients from one pool to another. The server adjusts the regular lease time so that the latest expiry time is at the given time + min-lease-time. A short min-lease-time enforces a step change, whereas a longer min-lease-time allows for a gradual change. The *time* is either a number of seconds since the Unix epoch, a UTC time string (e.g. `4 2007/08/24 09:14:32`), or a string with time zone offset in seconds (e.g. `4 2007/08/24 11:14:32 -7200`).

Reference: parameters

- The `adaptive-lease-time-threshold` statement:

  ```
  adaptive-lease-time-threshold percentage;
  ```

  When the number of allocated leases within a pool rises above the *percentage* given in this statement, the DHCP server decreases the lease length for new clients within this pool to `min-lease-time` seconds. Clients renewing an already valid (long) leases get at least the remaining time from the current lease. Since the leases expire faster, the server may either recover more quickly or avoid pool exhaustion

Copyright © 2014, QNX Software Systems Limited

entirely. Once the number of allocated leases drops below the threshold, the server reverts back to normal lease times. Valid percentages are between 1 and 99.

- The `always-broadcast` statement:

  ```
  always-broadcast flag;
  ```

  The DHCP and BOOTP protocols both require DHCP and BOOTP clients to set the broadcast bit in the flags field of the BOOTP message header. Unfortunately, some DHCP and BOOTP clients don't do this, and therefore may not receive responses from the DHCP server. The DHCP server can be made to always broadcast its responses to clients by setting this flag to `on` for the relevant scope; relevant scopes would be inside a conditional statement, as a parameter for a class, or as a parameter for a host declaration. To avoid creating excess broadcast traffic on your network, we recommend that you restrict the use of this option to as few clients as possible. For example, the Microsoft DHCP client is known not to have this problem, as are the OpenTransport and ISC DHCP clients.

- The `always-reply-rfc1048` statement:

  ```
  always-reply-rfc1048 flag;
  ```

  Some BOOTP clients expect RFC1048-style responses, but don't follow RFC1048 when sending their requests. You can tell that a client is having this problem if it isn't getting the options you have configured for it and if you see in the server log the message “(non-rfc1048)” printed with each BOOTREQUEST that is logged.

  If you want to send rfc1048 options to such a client, you can set the `always-reply-rfc1048` option in that client's host declaration, and the DHCP server will respond with an RFC-1048-style vendor options field. This flag can be set in any scope, and will affect all clients covered by that scope.

- The `authoritative` statement:

  ```
  authoritative;
  not authoritative;
  ```

  The DHCP server will normally assume that the configuration information about a given network segment isn't known to be correct and is not authoritative. This is so that if a naive user installs a DHCP server not fully understanding how to configure it, it doesn't send spurious DHCPNAK messages to clients that have obtained addresses from a legitimate DHCP server on the network. Network administrators setting up authoritative DHCP servers for their networks should always write `authoritative;` at the top of their configuration file to indicate that the DHCP server should send DHCPNAK messages to misconfigured clients. If this isn't done, clients will be unable to get a correct IP address after changing subnets until their old lease has expired, which could take quite a long time.

Copyright © 2014, QNX Software Systems Limited

Usually, writing `authoritative;` at the top level of the file should be sufficient. However, if a DHCP server is to be set up so that it is aware of some networks for which it is authoritative and some networks for which it isn't, it may be more appropriate to declare authority on a per-network-segment basis.

Note that the most specific scope for which the concept of authority makes any sense is the physical network segment; either a shared-network statement or a subnet statement that isn't contained within a shared-network statement. It isn't meaningful to specify that the server is authoritative for some subnets within a shared network, but not authoritative for others, nor is it meaningful to specify that the server is authoritative for some host declarations and not others.

- The `boot-unknown-clients` statement:

  ```
  boot-unknown-clients flag;
  ```

  If the `boot-unknown-clients` statement is present and has a value of `false` or `off`, then clients for which there is no `host` declaration will not be allowed to obtain IP addresses. If this statement is not present or has a value of `true` or `on`, then clients without host declarations will be allowed to obtain IP addresses, as long as those addresses aren't restricted by `allow` and `deny` statements within their `pool` declarations.

- The `db-time-format` statement:

  ```
  db-time-format [ default | local ] ;
  ```

  The DHCP server software outputs several timestamps when writing leases to persistent storage. This configuration parameter selects one of two output formats. The `default` format prints the day, date, and time in UTC, while the `local` format prints the system seconds-since-epoch, and helpfully provides the day and time in the system time zone in a comment. The time formats are described in detail in documentation for *`dhcpd.leases`* (p. 610).

- The `ddns-hostname` statement:

  ```
  ddns-hostname name;
  ```

  The *name* parameter should be the hostname that will be used in setting up the client's A and PTR records. If no `ddns-hostname` is specified in scope, then the server will derive the hostname automatically, using an algorithm that varies for each of the different update methods.

- The `ddns-domainname` statement:

  ```
  ddns-domainname name;
  ```

  The *name* parameter should be the domain name that will be appended to the client's hostname to form a fully-qualified domain-name (FQDN).

Copyright © 2014, QNX Software Systems Limited

- The `ddns-rev-domainname` statement:

  ```
  ddns-rev-domainname name;
  ```

  The *name* parameter should be the domain name that will be appended to the client's reversed IP address to produce a name for use in the client's PTR record. By default, this is `in-addr.arpa.`, but the default can be overridden here.

  The reversed IP address to which this domain name is appended is always the IP address of the client, in dotted quad notation, reversed; for example, if the IP address assigned to the client is `10.17.92.74`, then the reversed IP address is `74.92.17.10`. So a client with that IP address would, by default, be given a PTR record of `10.17.92.74.in-addr.arpa`.

- The `ddns-update-style` parameter:

  ```
  ddns-update-style style;
  ```

  The *style* parameter must be one of `ad-hoc`, `interim`, or `none`. The `ddns-update-style` statement is meaningful only in the outer scope — it's evaluated once after reading the `dhcpd.conf` file, rather than each time a client is assigned an IP address, so there is no way to use different DNS update styles for different clients. The default is `none`.

- The `ddns-updates` statement:

  ```
  ddns-updates flag;
  ```

  The `ddns-updates` parameter controls whether or not the server will attempt to do a DNS update when a lease is confirmed. Set this to `off` if the server shouldn't attempt to do updates within a certain scope. The `ddns-updates` parameter is on by default. To disable DNS updates in all scopes, it's preferable to use the `ddns-update-style` statement, setting the style to `none`.

- The `default-lease-time` statement:

  ```
  default-lease-time time;
  ```

  The *time* should be the length in seconds that will be assigned to a lease if the client requesting the lease doesn't ask for a specific expiration time. This is used for both DHCPv4 and DHCPv6 leases (it is also known as the "valid lifetime" in DHCPv6).

- The `do-forward-updates` statement:

  ```
  do-forward-updates flag;
  ```

  The `do-forward-updates` statement instructs the DHCP server as to whether it should attempt to update a DHCP client's A record when the client acquires or renews a lease. This statement has no effect unless DNS updates are enabled and

Copyright © 2014, QNX Software Systems Limited

`ddns-update-style` is set to `interim`. Forward updates are enabled by default. If this statement is used to disable forward updates, the DHCP server will never attempt to update the client's A record, and will only ever attempt to update the client's PTR record if the client supplies an FQDN that should be placed in the PTR record using the fqdn option. If forward updates are enabled, the DHCP server will still honor the setting of the `client-updates` flag.

- The `dynamic-bootp-lease-cutoff` statement:

  ```
  dynamic-bootp-lease-cutoff date;
  ```

  The `dynamic-bootp-lease-cutoff` statement sets the ending time for all leases assigned dynamically to BOOTP clients. Because BOOTP clients don't have any way of renewing leases, and don't know that their leases could expire, by default `dhcpd` assigns infinite leases to all BOOTP clients. However, it may make sense in some situations to set a cutoff date for all BOOTP leases — for example, the end of a school term, or the time at night when a facility is closed and all machines are required to be powered off.

  The *date* should be the date on which all assigned BOOTP leases will end. The date is specified in the form:

  W YYYY/MM/DD HH:MM:SS

  W is the day of the week expressed as a number from zero (Sunday) to six (Saturday). *YYYY* is the year, including the century. *MM* is the month expressed as a number from 1 to 12. *DD* is the day of the month, counting from 1. *HH* is the hour, from zero to 23. *MM* is the minute, and *SS* is the second. The time is always in Coordinated Universal Time (UTC), not local time.

- The `dynamic-bootp-lease-length` statement:

  ```
  dynamic-bootp-lease-length length;
  ```

  The `dynamic-bootp-lease-length` statement is used to set the length of leases dynamically assigned to BOOTP clients. At some sites, it may be possible to assume that a lease is no longer in use if its holder hasn't used BOOTP or DHCP to get its address within a certain time period. The period is specified in *length* as a number of seconds. If a client reboots using BOOTP during the timeout period, the lease duration is reset to *length*, so a BOOTP client that boots frequently enough will never lose its lease. Needless to say, this parameter should be adjusted with extreme caution.

- The `filename` statement:

  ```
  filename "file_name";
  ```


Copyright © 2014, QNX Software Systems Limited

The `filename` statement can be used to specify the name of the initial boot file which is to be loaded by a client. The *file_name* should be a filename recognizable to whatever file transfer protocol the client can be expected to use to load the file.

- The `fixed-address` declaration:

```
fixed-address address[, address ...];
```

  The `fixed-address` declaration is used to assign one or more fixed IP addresses to a client. It should appear only in a `host` declaration. If more than one address is supplied, then when the client boots, it will be assigned the address that corresponds to the network on which it is booting. If none of the addresses in the `fixed-address` statement are valid for the network to which the client is connected, that client will not match the `host` declaration containing that `fixed-address` declaration. Each *address* in the `fixed-address` declaration should be either an IP address or a domain name that resolves to one or more IP addresses.

- The `fixed-address6` declaration:

```
fixed-address6 ip6-address;
```

  The `fixed-address6` declaration is used to assign a fixed IPv6 addresses to a client. It should appear only in a `host` declaration.

- The `get-lease-hostnames` statement:

```
get-lease-hostnames flag;
```

  The `get-lease-hostnames` statement is used to tell `dhcpd` whether or not to look up the domain name corresponding to the IP address of each address in the lease pool and use that address for the DHCP `hostname` option. If *flag* is true, then this lookup is done for all addresses in the current scope. By default, or if *flag* is false, no lookups are done.

- The `hardware` statement:

```
hardware hardware-type hardware-address;
```

  In order for a BOOTP client to be recognized, its network hardware address must be declared using a `hardware` clause in the `host` statement. *hardware-type* must be the name of a physical hardware interface type. Currently, only the `ethernet` and `token-ring` types are recognized, although support for a `fddi` hardware type (and others) would also be desirable. The *hardware-address* should be a set of hexadecimal octets (numbers from 0 through FF) separated by colons. The `hardware` statement may also be used for DHCP clients.

- The `host-identifier option` statement:

```
host-identifier option option-name option-data;
```

Copyright © 2014, QNX Software Systems Limited

This identifies a DHCPv6 client in a `host` statement. The *option-name* is any option, and *option-data* is the value for the option that the client will send. The *option-data* must be a constant value.

- The `infinite-is-reserved` statement:

  ```
  infinite-is-reserved flag;
  ```

  ISC DHCP now supports "reserved" leases (see "*Reserved leases* (p. 608)," below). If this *flag* is on, the server will automatically reserve leases allocated to clients which requested an infinite (`0xFFFFFFFF`) lease-time. The default is off.

- The `lease-file-name` statement:

  ```
  lease-file-name name
  ```

  The *name* should be the name of the DHCP server's lease file. By default, this is `/var/db/dhcpd.leases`. This statement *must* appear in the outer scope of the configuration file; if it appears in some other scope, it will have no effect. Furthermore, it has no effect if overridden by the -lf flag or the ***PATH_DHCPD_DB*** environment variable.

- The `limit-addrs-per-ia` statement:

  ```
  limit-addrs-per-ia number;
  ```

  By default, the DHCPv6 server will limit clients to one IAADDR per IA option, meaning one address. If you wish to permit clients to hang onto multiple addresses at a time, configure a larger number here.

  Note that there is no present method to configure the server to forcibly configure the client with one IP address per each subnet on a shared network. This is left to future work.

- The `dhcpv6-lease-file-name` statement:

  ```
  dhcpv6-lease-file-name name;
  ```

  The *name* is the name of the lease file to use if and only if the server is running in DHCPv6 mode. By default, this is `/var/db/dhcpd6.leases`. This statement, like `lease-file-name`, *must* appear in the outer scope of the configuration file. It has no effect if overridden by the -lf flag or the ***PATH_DHCPD6_DB*** environment variable. If `dhcpv6-lease-file-name` isn't specified, but `lease-file-name` is, the latter value is used.

- The `local-port` statement:

  ```
  local-port port;
  ```

  This statement causes the DHCP server to listen for DHCP requests on the UDP port specified in *port*, rather than on port 67.

Copyright © 2014, QNX Software Systems Limited

- The `local-address` statement:

  ```
  local-address address;
  ```

  This statement causes the DHCP server to listen for DHCP requests sent to the specified *address*, rather than requests sent to all addresses. Since serving directly attached DHCP clients implies that the server must respond to requests sent to the all-ones IP address, this option can't be used if clients are on directly attached networks. It is realistically useful only for a server whose only clients are reached via unicasts, such as via DHCP relay agents.

  This statement is effective only if the server was compiled using the USE_SOCKETS `#define` statement, which is default on a small number of operating systems, and must be explicitly chosen at compile-time for all others. You can be sure if your server is compiled with USE_SOCKETS if you see lines of this format at startup:

  ```
  Listening on Socket/eth0
  ```

  Note also that since this *bind()*s all DHCP sockets to the specified address, that only one address may be supported in a daemon at a given time.

- The `log-facility` statement:

  ```
  log-facility facility;
  ```

  This statement causes the DHCP server to do all of its logging on the specified log facility once the `dhcpd.conf` file has been read. By default the DHCP server logs to the daemon facility. The default is `syslog` (if `syslogd` is running). Possible log facilities include `cron`, `ftp`, and `lpr`.

  Because the `log-facility` setting is controlled by the `dhcpd.conf` file, log messages printed while parsing the `dhcpd.conf` file or before parsing it are logged to the default log facility. To prevent this, see the README file included with this distribution, which describes how to change the default log facility. When this parameter is used, the DHCP server prints its startup message a second time after parsing the configuration file, so that the log will be as complete as possible.

- The `max-lease-time` statement:

  ```
  max-lease-time time;
  ```

  The *time* should be the maximum length in seconds that will be assigned to a lease. The only exception to this is that Dynamic BOOTP lease lengths, which aren't specified by the client, aren't limited by this maximum.

Copyright © 2014, QNX Software Systems Limited

- The `min-lease-time` statement:

  ```
  min-lease-time time;
  ```

  The *time* should be the minimum length in seconds that will be assigned to a lease.

- The `min-secs` statement

  ```
  min-secs seconds;
  ```

  The *seconds* variable should be the minimum number of seconds since a client began trying to acquire a new lease before the DHCP server will respond to its request. The number of seconds is based on what the client reports, and the maximum value that the client can report is 255 seconds. Generally, setting this to one will result in the DHCP server not responding to the client's first request, but always responding to its second request.

  This can be used to set up a secondary DHCP server that never offers an address to a client until the primary server has been given a chance to do so. If the primary server is down, the client will bind to the secondary server, but otherwise clients should always bind to the primary. Note that this doesn't, by itself, permit a primary server and a secondary server to share a pool of dynamically allocatable addresses.

- The `next_server` statement:

  ```
  next_server server-name;
  ```

  The `next_server` statement is used to specify the host address of the server from which the initial boot file (specified in the `filename` statement) is to be loaded. The *server-name* should be a numeric IP address or a domain name.

- The `omapi-port` statement:

  ```
  omapi-port port;
  ```

  The `omapi-port` statement causes the DHCP server to listen for OMAPI connections on the specified port. This statement is required to enable the OMAPI protocol, which is used to examine and modify the state of the DHCP server as it is running.

- The `one-lease-per-client` statement:

  ```
  one-lease-per-client flag;
  ```

  If this flag is enabled, whenever a client sends a DHCPREQUEST for a particular lease, the server will automatically free any other leases the client holds. This presumes that when the client sends a DHCPREQUEST, it has forgotten any lease not mentioned in the DHCPREQUEST — i.e., the client has only a single network interface *and* it doesn't remember leases it's holding on networks to which it isn't

Copyright © 2014, QNX Software Systems Limited

currently attached. Neither of these assumptions is guaranteed or provable, so we urge caution in the use of this statement.

- The `pid-file-name` statement:

  ```
  pid-file-name name;
  ```

  The *name* should be the name of the DHCP server's process ID file. This is the file in which the DHCP server's process ID is stored when the server starts. By default, this is `/var/run/dhcpd.pid`. Like the `lease-file-name` statement, this statement must appear in the outer scope of the configuration file. It has no effect if overridden by the -pf flag or the ***PATH_DHCPD_PID*** environment variable.

- The `dhcpv6-pid-file-name` statement:

  ```
  dhcpv6-pid-file-name name;
  ```

  The *name* is the name of the pid file to use if and only if the server is running in DHCPv6 mode. By default, this is `/var/db/dhcpd6.pid`. This statement, like `pid-file-name`, *must* appear in the outer scope of the configuration file. It has no effect if overridden by the -pf flag or the ***PATH_DHCPD6_PID*** environment variable. If `dhcpv6-pid-file-name` isn't specified, but `pid-file-name` is, the latter value will be used.

- The `ping-check` statement:

  ```
  ping-check flag;
  ```

  When the DHCP server is considering dynamically allocating an IP address to a client, it first sends an ICMP Echo request (a *ping*) to the address being assigned. It waits for a second, and if no ICMP Echo response has been heard, it assigns the address. If a response *is* heard, the lease is abandoned, and the server doesn't respond to the client.

  This `ping-check` introduces a default one-second delay in responding to DHCPDISCOVER messages, which can be a problem for some clients. The default delay of one second may be configured using the `ping-timeout` parameter. The `ping-check` configuration parameter can be used to control checking; if its value is false, no ping check is done.

- The `ping-timeout` statement:

  ```
  ping-timeout seconds;
  ```

  If the DHCP server determined it should send an ICMP echo request (a *ping*) because the ping-check statement is true, `ping-timeout` allows you to configure how many seconds the DHCP server should wait for an ICMP Echo response to be heard, if no ICMP Echo response has been received before the timeout expires, it assigns the address. If a response *is* heard, the lease is abandoned, and the server

Copyright © 2014, QNX Software Systems Limited

doesn't respond to the client. If no value is set, `ping-timeout` defaults to 1 second.

- The `preferred-lifetime` statement:

  ```
  preferred-lifetime seconds;
  ```

  IPv6 addresses have “valid” and “preferred” lifetimes. The valid lifetime determines at what point at lease might be said to have expired, and is no longer useable. A preferred lifetime is an advisory condition to help applications move off of the address and onto currently valid addresses (should there still be any open TCP sockets or similar).

  The preferred lifetime defaults to the renew+rebind timers, or 3/4 the default lease time if none were specified.

- The `remote-port` statement:

  ```
  remote-port port;
  ```

  This statement causes the DHCP server to transmit DHCP responses to DHCP clients upon the UDP port specified in *port*, rather than on port 68. In the event that the UDP response is transmitted to a DHCP Relay, the server generally uses the `local-port` configuration value. Should the DHCP Relay happen to be addressed as 127.0.0.1, however, the DHCP Server transmits its response to the `remote-port` configuration value. This is generally useful only for testing purposes, and this configuration value should generally not be used.

- The `server-identifier` statement:

  ```
  server-identifier hostname;
  ```

  The `server-identifier` statement can be used to define the value that's sent in the DHCP Server Identifier option for a given scope. The value specified *must* be an IP address for the DHCP server, and must be reachable by all clients served by a particular scope.

  The use of the `server-identifier` statement isn't recommended; the only reason to use it is to force a value other than the default value to be sent on occasions where the default value would be incorrect. The default value is the first IP address associated with the physical network interface on which the request arrived.

  The usual case where the `server-identifier` statement needs to be sent is when a physical interface has more than one IP address, and the one being sent by default isn't appropriate for some or all clients served by that interface. Another common case is when an alias is defined for the purpose of having a consistent IP address for the DHCP server, and it is desired that the clients use this IP address when contacting the server.

Copyright © 2014, QNX Software Systems Limited

Supplying a value for the `dhcp-server-identifier` option is equivalent to using the `server-identifier` statement.

- The `server-duid` statement:

```
server-duid LLT [ hardware-type timestamp hardware-address ] ;
server-duid EN enterprise-number enterprise-identifier ;
server-duid LL [ hardware-type hardware-address ] ;
```

The `server-duid` statement configures the server DHCP Unique Identifier (DUID). You may pick LLT (link local address plus time), EN (enterprise), or LL (link local).

If you choose LLT or LL, you may specify the exact contents of the DUID. Otherwise the server will generate a DUID of the specified type.

If you choose EN, you must include the enterprise number and the *enterprise-identifier*.

The default `server-duid` type is LLT.

- The `server-name` statement:

```
server-name name;
```

The `server-name` statement can be used to inform the client of the name of the server from which it is booting. The *name* should be the one that will be provided to the client.

- The `site-option-space` statement:

```
site-option-space name;
```

The `site-option-space` statement can be used to determine from what option space site-local options will be taken. This can be used in much the same way as the `vendor-option-space` statement. Site-local options in DHCP are those options whose numeric codes are greater than 224. These options are intended for site-specific uses, but are frequently used by vendors of embedded hardware that contains DHCP clients. Because site-specific options are allocated on an ad hoc basis, it is quite possible that one vendor's DHCP client might use the same option code that another vendor's client uses, for different purposes. The `site-option-space` option can be used to assign a different set of site-specific options for each such vendor, using conditional evaluation (for details, see the *DHCP Conditional Evaluation* (p. 505) entry).

- The `stash-agent-options` statement:

```
stash-agent-options flag;
```

If the `stash-agent-options` parameter is true for a given client, the server will record the relay agent information options sent during the client's initial

Copyright © 2014, QNX Software Systems Limited

DHCPREQUEST message when the client was in the SELECTING state and behave as if those options are included in all subsequent DHCPREQUEST messages sent in the RENEWING state. This works around a problem with relay agent information options, which is that they usually not appear in DHCPREQUEST messages sent by the client in the RENEWING state, because such messages are unicast directly to the server and not sent through a relay agent.

- The `update-conflict-detection` statement:

  ```
  update-conflict-detection flag;
  ```

  If the `update-conflict-detection` parameter is `true`, the server will perform standard DHCID multiple-client, one-name conflict detection. If the parameter has been set to `false`, the server will skip this check and instead simply tear down any previous bindings to install the new binding without question. The default is `true`.

- The `update-optimization` statement:

  ```
  update-optimization flag;
  ```

  If the `update-optimization` parameter is false for a given client, the server will attempt a DNS update for that client each time the client renews its lease, rather than only attempting an update when it appears to be necessary. This will allow the DNS to heal from database inconsistencies more easily, but the cost is that the DHCP server must do many more DNS updates. We recommend leaving this option enabled, which is the default. This option affects only the behavior of the interim DNS update scheme, and has no effect on the ad-hoc DNS update scheme. If this parameter isn't specified or is true, the DHCP server will update only when the client information changes, the client gets a different lease, or the client's lease expires.

- The `update-static-leases` statement:

  ```
  update-static-leases flag;
  ```

  The `update-static-leases` flag, if enabled, causes the DHCP server to do DNS updates for clients even if those clients are being assigned their IP address using a `fixed-address` statement; that is, the client is being given a static assignment. This can work only with the `interim` DNS update scheme. It isn't recommended because the DHCP server has no way to tell that the update has been done, and therefore will not delete the record when it isn't in use. Also, the server must attempt the update each time the client renews its lease, which could have a significant performance impact in environments that place heavy demands on the DHCP server.

Copyright © 2014, QNX Software Systems Limited

- The `use-host-decl-names` statement:

  ```
  use-host-decl-names flag;
  ```

  If the `use-host-decl-names` parameter is true in a given scope, then for every host declaration within that scope, the name provided for the host declaration will be supplied to the client as its hostname. So, for example,

  ```
  group {
     use-host-decl-names on;

     host joe {
        hardware ethernet 08:00:2b:4c:29:32;
        fixed-address joe.fugue.com;
     }
  }
  ```

  is equivalent to:

  ```
  host joe {
     hardware ethernet 08:00:2b:4c:29:32;
     fixed-address joe.fugue.com;
     option host-name "joe";
  }
  ```

  An `option hostname` statement within a host declaration will override the use of the name in the host declaration.

  It should be noted here that most DHCP clients completely ignore the host-name option sent by the DHCP server, and there is no way to configure them not to do this. So you generally have a choice of either not having any hostname to client IP address mapping that the client will recognize, or doing DNS updates. It is beyond the scope of this document to describe how to make this determination.

- The `use-lease-addr-for-default-route` statement:

  ```
  use-lease-addr-for-default-route flag;
  ```

  If the `use-lease-addr-for-default-route` parameter is true in a given scope, then instead of sending the value specified in the routers option (or sending no value at all), the IP address of the lease being assigned is sent to the client. This supposedly causes Win95 machines to ARP for all IP addresses, which can be helpful if your router is configured for proxy ARP. The use of this feature isn't recommended, because it won't work for many DHCP clients.

- The `vendor-option-space` statement:

  ```
  vendor-option-space string;
  ```

  The `vendor-option-space` parameter determines from what option space vendor options are taken. The use of this configuration parameter is illustrated in "*Vendor-encapsulated options* (p. 541)" in the DHCP options entry.

Setting parameter values using expressions

Copyright © 2014, QNX Software Systems Limited

Sometimes it's helpful to be able to set the value of a DHCP server parameter based on some value that the client has sent. To do this, you can use expression evaluation. For information on writing expressions, see the *DHCP Conditional Evaluation* (p. 505) entry. To assign the result of an evaluation to an option, define the option as follows:

my-parameter = *expression* ;

For example:

```
ddns-hostname = binary-to-ascii (16, 8, "-",
                                 substring (hardware, 1, 6));
```

Reserved leases

It's often useful to allocate a single address to a single client, in approximate perpetuity. Host statements with `fixed-address` clauses exist to a certain extent to serve this purpose, but because host statements are intended to approximate "static configuration", they suffer from not being referenced in a litany of other Server Services, such as dynamic DNS, failover, "on events" and so forth.

If a standard dynamic lease, as from any range statement, is marked `reserved`, then the server will allocate this lease only to the client it is identified by (be that by client identifier or hardware address).

In practice, this means that the lease follows the normal state engine, enters ACTIVE state when the client is bound to it, expires, or is released, and any events or services that would normally be supplied during these events are processed normally, as with any other dynamic lease. The only difference is that failover servers treat reserved leases as special when they enter the FREE or BACKUP states; each server applies the lease into the state it may allocate from, and the leases aren't placed on the queue for allocation to other clients. Instead they may only be "found" by client identity. The result is that the lease is only offered to the returning client.

Care should probably be taken to ensure that the client has only one lease within a given subnet that it is identified by.

Leases may be set `reserved` either through OMAPI, or through the `infinite-is-reserved` configuration option (if this is applicable to your environment and mixture of clients).

It should also be noted that leases marked `reserved` are effectively treated the same as leases marked `bootp`.

See also:

- For information about DHCP option statements, see the *DHCP options* (p. 514) entry.
- For information about expressions used in DHCP option statements and elsewhere, see the *DHCP Conditional Evaluation* (p. 505) entry.
- *RFC2132*, *RFC2131*

Copyright © 2014, QNX Software Systems Limited

Contributing author:

dhcpd.conf was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Information about Internet Systems Consortium can be found at *http://www.isc.org*.

Copyright © 2014, QNX Software Systems Limited

dhcpd.leases, dhcpd6.leases

DHCP server database of assigned leases

Name:

- `/var/db/dhcpd.leases` for DHCPv4
- `/var/db/dhcpd6.leases` for DHCPv6

Description:

The Internet Systems Consortium DHCP Server keeps a persistent database of leases that it has assigned. This database is a free-form ASCII file containing a series of lease declarations. Every time a lease is acquired, renewed or released, its new value is recorded at the end of the lease file. So if more than one declaration appears for a given lease, the last one in the file is the current one.

When *dhcpd* (p. 552) is first installed, there is no lease database. However, `dhcpd` requires that a lease database be present before it will start. To make the initial lease database, just create an empty file called `/var/db/dhcpd.leases` for DHCPv4, or `/var/db/dhcpd6.leases` for DHCPv6. You can do this with:

```
touch /var/db/dhcpd.leases
```

In order to prevent the lease database from growing without bound, the file is rewritten from time to time. First, a temporary lease database is created, and all known leases are dumped to it. Then, the old lease database is renamed `/var/db/dhcpd.leases~` (`/var/db/dhcpd6.leases~` for DHCPv6). Finally, the newly written lease database is moved into place.

Format

Lease descriptions are stored in a format that is parsed by the same recursive descent parser used to read the *dhcpd.conf* (p. 566) and *dhclient.conf* (p. 492) files. Lease files can contain lease declarations, and also group and subgroup declarations, host declarations and failover state declarations. Group, subgroup and host declarations are used to record objects created using the OMAPI protocol.

The lease file is a log-structured file; whenever a lease changes, the contents of that lease are written to the end of the file. This means that it is entirely possible and quite reasonable for there to be two or more declarations of the same lease in the lease file at the same time. In that case, the instance of that particular lease that appears last in the file is the one that is in effect.

Group, subgroup and host declarations in the lease file are handled in the same manner, except that if any of these objects are deleted, a `rubout` is written to the lease file. This is just the same declaration, with `{ deleted; }` in the scope of the

Copyright © 2014, QNX Software Systems Limited

declaration. When the lease file is rewritten, any such rubouts that can be eliminated are eliminated. It's possible to delete a declaration in the `dhcpd.conf` file; in this case, the rubout can never be eliminated from the `dhcpd.leases` file.

The lease declaration

A lease declaration takes this form:

```
lease ip-address { statements... }
```

Each lease declaration includes the single IP address that has been leased to the client. The statements within the braces define the duration of the lease and to whom it is assigned.

```
starts date;
ends date;
tstp date;
tsfp date;
atsfp date;
cltt date;
```

The start and end time of a lease are recorded using the `starts` and `ends` statements. The `tstp` statement is specified if the failover protocol is being used, and indicates what time the peer has been told the lease expires. The `tsfp` statement is also specified if the failover protocol is being used, and indicates the lease expiry time that the peer has acknowledged. The `atsfp` statement is the actual time sent from the failover partner. The `cltt` statement is the client's last transaction time.

The *date* is specified in two ways, depending on the configuration value for the `db-time-format` parameter:

- If it was set to `default`, then the *date* fields appear as follows:

  ```
  weekday year/month/day hour:minute:second
  ```

  The weekday is present to make it easy for a human to tell when a lease expires — it's specified as a number from zero to six, with zero being Sunday. The day of week is ignored on input. The year is specified with the century, so it should generally be four digits except for really long leases. The month is specified as a number starting with 1 for January. The day of the month is likewise specified starting with 1. The hour is a number between 0 and 23, the minute a number between 0 and 59, and the second also a number between 0 and 59.

  Lease times are specified in Universal Coordinated Time (UTC), not in the local time zone. There is probably nowhere in the world where the times recorded on a lease are always the same as wall clock times. On most Unix machines, you can display the current time in UTC by typing `date -u`.

- If the `db-time-format` was configured to `local`, then the *date* fields appear as follows:

  ```
  epoch  seconds-since-epoch;  #  day-name month-name day-number
  hours:minutes:seconds year
  ```

Copyright © 2014, QNX Software Systems Limited

The *seconds-since-epoch* is as according to the system's local clock (often referred to as “Unix time”). The # symbol supplies a comment that describes what actual time this is, according to the system's configured time zone, at the time the value was written. It is provided only for human inspection.

- If a lease will never expire, *date* is `never` instead of an actual date.

The other statements include the following:

`hardware` *hardware-type mac-address*;

Records the MAC address of the network interface on which the lease will be used. It is specified as a series of hexadecimal octets, separated by colons.

`uid` *client-identifier*;

Records the client identifier used by the client to acquire the lease. Clients are not required to send client identifiers, and this statement appears only if the client did in fact send one. Client identifiers are normally an ARP type (1 for ethernet) followed by the MAC address, just like in the `hardware` statement, but this is not required.

The client identifier is recorded as a colon-separated hexadecimal list or as a quoted string. If it is recorded as a quoted string and it contains one or more non-printable characters, those characters are represented as octal escapes - a backslash character followed by three octal digits.

`client-hostname` *hostname* ;

Most DHCP clients will send their hostname in the `host-name` option. If a client sends its hostname in this way, the hostname is recorded on the lease with a `client-hostname` statement. This is not required by the protocol, however, so many specialized DHCP clients do not send a host-name option.

`abandoned`;

Indicates that the DHCP server has abandoned the lease. In that case, the `abandoned` statement will be used to indicate that the lease should not be reassigned. For information about abandoned leases, see the documentation for *`dhcpd.conf`* (p. 566).

`binding state` *state*; `next binding state` *state*;

The lease's binding state. When the DHCP server is not configured to use the failover protocol, a lease's binding state will be either `active` or `free`. The failover protocol adds some additional transitional states, as well as the

Copyright © 2014, QNX Software Systems Limited

`backup` state, which indicates that the lease is available for allocation by the failover secondary.

The `next binding state` statement indicates what state the lease will move to when the current state expires. The time when the current state expires is specified in the `ends` statement.

`option agent.circuit-id` ***string*****; `option agent.remote-id`** ***string*****;**

The `option agent.circuit-id` and `option agent.remote-id` statements are used to record the circuit ID and remote ID options send by the relay agent, if the relay agent uses the `relay agent information` option. This allows these options to be used consistently in conditional evaluations even when the client is contacting the server directly rather than through its relay agent.

`set` ***variable*** **=** ***value*****;**

Sets the value of a variable on the lease. For general information on variables, see the *DHCP Conditional Evaluation* (p. 505) entry.

The variables include:

- `ddns-text` — used to record the value of the client's TXT identification record when the interim DDNS update style has been used to update the DNS for a particular lease.
- `ddns-fwd-name` — records the value of the name used in updating the client's A record if a DDNS update has been successfully done by the server. The server may also have used this name to update the client's PTR record.
- `ddns-client-fqdn` — if the server is configured to use the interim DDNS update style, and is also configured to allow clients to update their own FQDNs, and the client did in fact update its own FQDN, then the `ddns-client-fqdn` variable records the name that the client has indicated it is using. This is the name that the server will have used to update the client's PTR record in this case.
- `ddns-rev-name` — if the server successfully updates the client's PTR record, this variable will record the name that the DHCP server used for the PTR record. The name to which the PTR record points will be either the `ddns-fwd-name` or the `ddns-client-fqdn`.

`on` ***events*** **{** ***statements...*** **}**

A list of statements to execute if a certain event occurs. The possible events that can occur for an active lease are `release` and `expiry`. More than one event can be specified; if so, the events are separated by `|` characters.

Copyright © 2014, QNX Software Systems Limited

bootp; reserved;

These two statements are effectively flags. If present, they indicate that the BOOTP and RESERVED failover flags, respectively, should be set. BOOTP and RESERVED dynamic leases are treated differently than normal dynamic leases, as they may only be used by the client to which they are currently allocated.

The failover peer state declaration

The state of any failover peering arrangements is also recorded in the lease file, using the `failover peer` statement:

```
failover peer name state {
my state state at date;
peer state state at date;
}
```

The states of the peer named *name* is being recorded. Both the state of the running server (`my state`) and the other failover partner (`peer` *state*) are recorded. The following states are possible:

- `unknown-state`
- `partner-down`
- `normal`
- `communications-interrupted`
- `resolution-interrupted`
- `potential-conflict`
- `recover`
- `recover-done`
- `shutdown`
- `paused`
- `startup`

Contributing author:

`dhcpd` was written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Information about Internet Systems Consortium can be found at: *http://www.isc.org/*.

See also:

RFC2132, *RFC2131*

Copyright © 2014, QNX Software Systems Limited

dhcrelay

Dynamic Host Configuration Protocol relay agent

Syntax:

```
dhcrelay [-4] [-dqaD] [-p port] [-c count]
         [-A length] [-pf pid-file ] [--no-pid]
         [-m append | replace | forward | discard]
         -i interface0 [... -i interfaceN]
         server0 [ ...serverN ]

dhcrelay -6 [-dqI] [-p port] [-c count]
         [-pf pid-file ] [--no-pid]
         -l lower0 [... -l lowerN]
         -u upper0 [... -u upperN]
```

Runs on:

QNX Neutrino

Options:

Protocol-selection options:

-4

Run `dhcrelay` as a DHCPv4/BOOTP relay agent. This is the default mode of operation, so the option isn't necessary, but you can specify it for clarity. It's incompatible with -6.

-6

Run `dhcrelay` as a DHCPv6 relay agent. Incompatible with the -4 option. If you specify this option, the names of the default files include a "6":

IPv4 name	IPv6 name
`/var/run/dhcrelay.pid`	`/var/run/dhcrelay6.pid`

Specifying DHCPv4/BOOTP servers:

In DHCPv4 mode, you must specify a list of one or more server addresses to which DHCP/BOOTP queries should be relayed.

Options available for both DHCPv4 and DHCPv6:

-c *count*

Copyright © 2014, QNX Software Systems Limited

The maximum hop count. When forwarding packets, `dhcrelay` discards packets that have reached a hop count of *count*. The default is 10, and the maximum is 255.

-d

Force `dhcrelay` to run as a foreground process. Useful when running `dhcrelay` under a debugger.

-p *port*

Listen and transmit on the specified port. This is mostly useful for debugging purposes. The default is port 67 for DHCPv4/BOOTP, or port 547 for DHCPv6.

-q

Quiet mode; prevents `dhcrelay` from printing its network configuration on startup.

-pf *pid-file*

The path to the alternate pid file.

--no-pid

Disable the writing of pid files. By default the program writes a pid file.

Options available in DHCPv4 mode only:

-a

Append an agent option field to each request before forwarding it to the server. Agent option fields in responses sent from servers to clients are stripped before forwarding such responses back to the client. The agent option field contains two agent options: the Circuit ID suboption and the Remote ID suboption. Currently, the Circuit ID is the printable name of the interface on which the client request was received. The client supports inclusion of a Remote ID suboption as well, but this isn't used by default.

-A *length*

Specify the maximum packet size to send to a DHCPv4/BOOTP server. This might be done to allow sufficient space for the addition of relay agent options while still fitting into the Ethernet MTU size.

-D

Drop packets from upstream servers if they contain Relay Agent Information options that indicate that they were generated in response to a query that

Copyright © 2014, QNX Software Systems Limited

came via a different relay agent. If you don't specify this option, such packets are relayed anyway.

-i *ifname*

Listen for DHCPv4/BOOTP queries on interface *ifname*. You can specify multiple interfaces by using more than one -i option. If you don't specify any interfaces, `dhcrelay` identifies all network interfaces, eliminating non-broadcast interfaces if possible, and attempts to listen on all of them.

-m append | replace | forward | discard

Control the handling of incoming DHCPv4 packets that already contain relay agent options. If such a packet doesn't have *giaddr* set in its header, the DHCP standard requires that the packet be discarded. However, if *giaddr* is set, the relay agent may handle the situation in four ways:

- It may append its own set of relay options to the packet, leaving the supplied option field intact.
- It may replace the existing agent option field.
- It may forward the packet unchanged.
- It may discard it.

Options available in DHCPv6 mode only:

-l

Force the use of the DHCPv6 Interface-ID option. This option is automatically sent when there are two or more downstream interfaces in use, to disambiguate between them. The -i option causes `dhcrelay` to send the option even if there is only one downstream interface.

-l [*address*%]*ifname*[#*index*]

("el") Specifies the "lower" network interface for DHCPv6 relay mode: the interface on which queries will be received from clients or from other relay agents. You must include at least one -l option in the command line when you're running in DHCPv6 mode. The interface name *ifname* is a mandatory parameter. You can specify the link address by *address*%; if you don't, `dhcrelay` uses the first non-link-local address configured on the interface. The optional #*index* parameter specifies the interface index.

-u [*address*%]*ifname*

Specifies the "upper" network interface for DHCPv6 relay mode: the interface to which queries from clients and other relay agents should be forwarded. You must include at least one -u option in the command line when you're

Copyright © 2014, QNX Software Systems Limited

running in DHCPv6 mode. The interface name *ifname* is a mandatory parameter. You can specify the destination unicast or multicast address by *address*%; if you don't, the relay agent forwards to the DHCPv6 All DHCP Relay Agents and Servers multicast address.

It's possible to specify the same interface with different addresses more than once, and even, when the system supports it, to use the same interface as both upper and lower interfaces.

Description:

The Internet Systems Consortium DHCP Relay Agent, `dhcrelay`, provides a means for relaying DHCP and BOOTP requests from a subnet to which no DHCP server is directly connected to one or more DHCP servers on other subnets. It supports both DHCPv4/BOOTP and DHCPv6 protocols.

The DHCP Relay Agent listens for DHCPv4 or DHCPv6 queries from clients or other relay agents on one or more interfaces, passing them along to "upstream" servers or relay agents as specified on the command line. When a reply is received from upstream, it's multicast or unicast back downstream to the source of the original request.

See also:

RFC3315, *RFC2132*, *RFC2131*

Files:

The `dhcrelay` agent depends on the following libraries:

- `libcrypto.so`
- `libsocket.so`
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)

Caveats:

Using the same interface on both upper and lower sides may cause loops, so when you're running `dhcrelay` this way, you should set the maximum hop count to a low value.

The loopback interface isn't (yet) recognized as a valid interface.

Copyright © 2014, QNX Software Systems Limited

diff

Compare two files, line by line (GNU)

Syntax:

```
diff [option] file1 file2
```

Runs on:

QNX Neutrino

Options:

-i or --ignore-case

Ignore case differences in the contents of the files.

--ignore-file-name-case

Ignore case when comparing filenames.

--no-ignore-file-name-case

Consider case when comparing filenames.

-E or --ignore-tab-expansion

Ignore changes due to tab expansion.

-b or --ignore-space-change

Ignore changes in the amount of white space.

-w or --ignore-all-space

Ignore all white space.

-B or --ignore-blank-lines

Ignore changes whose lines are all blank.

-I *RE* or --ignore-matching-lines=*RE*

Ignore changes whose lines all match the given regular expression.

--strip-trailing-cr

Strip trailing carriage returns from the input.

-a or --text

Copyright © 2014, QNX Software Systems Limited

Treat all files as text.

-c or -C *num* or --context[=*num*]

Output *num* (default 3) lines of copied context.

-u or -U *num* or --unified[=*num*]

Output *num* (default 3) lines of unified context.

--label *label*

Use *label* instead of the filename in the output. You can specify this option twice; the first applies to *file1*, and the second to *file2*.

-p or --show-c-function

Show which C function each change is in.

-F *RE* or --show-function-line=*RE*

Show the most recent line that matches the given regular expression.

-q or --brief

Output only whether or not the files differ.

-e or --ed

Output an ed script.

--normal

Display the differences in the normal format.

-n or --rcs

Display the differences in RCS format.

-y or --side-by-side

Display the output in two columns.

-W *num* or --width=*num*

Output at most *num* (default 130) print columns.

--left-column

Output only the left column of common lines.

--suppress-common-lines

Don't display any lines that are common to both files.

-D *name* or --ifdef=*name*

Copyright © 2014, QNX Software Systems Limited

Output merged file to show `#ifdef` *name* differences.

--*GTYPE*-group-format=*GFMT*

Use the given format to output groups of lines in an if-then-else format. *GTYPE* can be `old`, `new`, `changed`, or `unchanged`.

GFMT may contain:

%<

Lines from *file1*.

%>

Lines from *file2*.

%=

Lines common to *file1* and *file2*.

%[-][*width*][.[*prec*]]{doxX}*letter*

Use the given *printf*-style specification for the given *letter*. The letters are as follows for the new group; use lowercase letters for the old group:

- `F` — the first line number
- `L` — the last line number
- `N` — the number of lines = L-F+1
- `E` — F-1
- `M` — L+1.

%%

A literal %.

%c '*C*'

The single character, *C*.

%c '*OOO*'

The character with the given octal code, *OOO*.

--line-format=*LFMT*

Use the given format to output all input lines in an if-then-else format.

--*LTYPE*-line-format=*LFMT*

Copyright © 2014, QNX Software Systems Limited

Use the given format to output individual lines in an if-then-else format. *LTYPE* can be `old`, `new`, or `unchanged`. *LFMT* may contain:

%L

Contents of the line.

%l

Contents of the line, excluding any trailing newline.

%[-][*width*][.[*prec*]]{doxX}n

Use the given *printf*-style specification for input line numbers.

%%

A literal %.

%c '*C*'

The single character, *C*.

%c '*OOO*'

The character with the given octal code, *OOO*.

-l or --paginate

Pass the output through *pr* (p. 1571) to paginate it.

-t or --expand-tabs

Expand tabs into spaces in the output.

-T or --initial-tab

Make tabs line up by prepending a tab.

-r or --recursive

Recursively compare any subdirectories found.

-N or --new-file

Treat absent files as being empty.

--unidirectional-new-file

Treat absent first files as being empty.

-s or --report-identical-files

Report when two files are the same.

Copyright © 2014, QNX Software Systems Limited

-x *pattern* or --exclude=*pattern*

Exclude files that match the given pattern.

-X *file* or --exclude-from=*file*

Exclude files that match any pattern in *file*.

-S *file* or --starting-file=*file*

Start with *file* when comparing directories.

--from-file=*file1*

Compare *file1* to all operands. The *file1* argument can also be the name of a directory.

--to-file=*file2*

Compare all operands to *file2*. The *file2* argument can also be the name of a directory.

--horizon-lines=*num*

Keep *num* lines of the common prefix and suffix.

-d or --minimal

Try hard to find a smaller set of changes.

--speed-large-files

Assume large files and many scattered small changes.

-v or --version

Output version information.

--help

Display a help message.

file1*, *file2

Pathnames of the files to be compared. These can be in the following forms:

- *file1 file2*
- *directory1 directory2*
- *directory file...*
- *file... directory*

Copyright © 2014, QNX Software Systems Limited

If you specify the --from-file or --to-file option, there are no restrictions on the files. If you specify a dash (`-`) instead of a filename, `diff` reads from standard input.

Description:

The `diff` utility reports the differences between two files.

If you use `diff` to compare binary files, `diff` simply reports whether or not the files are different. If you want to see the differences between two binary files, use *cmp* (p. 134).

For any two files, there may be several correct interpretations of the differences between them. The `diff` utility attempts to generate the smallest number of additions, changes, and deletions required to convert *file1* into *file2*. No output is produced if the files are identical.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

For detailed documentation about `diff`, see the GNU website at *http://www.gnu.org/*.

Exit status:

0

No differences were found.

1

Differences were found.

>1

An error occurred.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

dig

DNS domain information groper lookup utility

Syntax:

```
dig [@server] [-b address] [-c class] [-f filename]
    [-k filename] [-p port_num] [-q name] [-t type]
    [-x addr] [-y [hmac:]name:key] [-4] [-6] [name]
    [type] [class] [queryopt...]

dig [-h]

dig [global-queryopt...] [query...]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?dig++NetBSD-5.0* in the NetBSD documentation.

Description:

The `dig` (domain information groper) utility is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?dig++NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

dinit

Initialize a disk for use as a QNX 4 filesystem (QNX)

Syntax:

```
dinit [-8bpqr] [-F|h] [-B filename|-O] [-d drive_number]
      [-f bootfile] [-i blocks] [-L label|-l label]
      [-m message] [-N] [-R] [-r] [-S size] drive
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-8

Use the int 13 extended loader for disks bigger than 8.4Gig.

-B *filename*

Use the 512-byte OS loader from this file instead of the standard QNX Neutrino RTOS loader. Note that on a partitioned hard disk, the OS loader is the second loader, not the primary bootstrap loader, which is written to the first block of the hard disk by the *fdisk* (p. 734) utility. On a nonpartitioned device, (e.g. floppy) the OS loader is the primary (and only) bootstrap loader.

-b

Don't initialize the filesystem; just write the OS loader to disk. You can use this option with -m, or -O.

-d *drive_number*

The BIOS drive number to use for booting the second stage loader (`diskpc2`). This enables you to set up the loader to boot when the drive is configured as either a primary or secondary drive. Common values for *drive_number* are: `00` for the first floppy drive, `80` for the first hard drive, and `81` for the second hard drive.

This option is required for booting from secondary hard drives (if you specify an explicit drive number, it overrides -F or -h).

-F

Initialize a floppy or LS120 disk.

Copyright © 2014, QNX Software Systems Limited

-f *bootfile*

Write the specified operating system boot image to the `/.boot` file on the newly initialized disk.

-H or -h

Initialize a hard or compact flash disk. You can't initialize a hard disk unless you specify this option.

-i *blocks*

The initial size, in blocks, of the `.inodes` file. The default is 16. You don't usually have to change this setting; for more information, see Fine-Tuning Your System in the QNX Neutrino *User's Guide*.

-l *label*

("el") Write the given volume label to disk after initializing.

-L *label*

Write only the given volume label to disk. You can remove a label by using the -L option with an empty string. For example:

```
dinit -L "" /dev/fd0
```

-m *message*

Replace the message the OS displays when booting from disk with *message*.

-N

Don't create support for long filenames (more than 48 characters) on this new filesystem.

To add support for long filenames to an existing QNX 4 filesystem, log in as `root` and create an empty, read-only (permissions 0444) file named `.longfilenames` in the root directory of the filesystem.

-O

Use the old QNX bootstrap loader. The old loader loads at (real mode) `0x60:0`, always. The newer loader looks for a signature byte in the beginning of the OS image to determine if it's old or new, and loads at `0x60:00` or `0x80:00` as appropriate. The start address for new images is `0x0` relative to the load address, while the start address for old images is `0x20` relative to the load address.

-p

Copyright © 2014, QNX Software Systems Limited

Pause for a keystroke before continuing.

-q

Be quiet; don't echo or question.

-R

Create a `.diskroot` file in the `root` directory.

-r

Write only the root block to disk; see "*Caveats* (p. 630)," below.

-S *size*

When used on a file, grow it to this size, which can include a suffix of `k`, `m`, or `g`.

drive

The drive on which to initialize the hard disk or diskette (e.g. `/dev/fd0`, `/dev/hd0t77`).

Device names under Windows differ from those under QNX operating systems. For example, under QNX Neutrino:

```
dinit -f hello.ifs /dev/fd0
```

Under Windows:

```
dinit -f hello.ifs a:
```

Description:

The `dinit` utility initializes a formatted diskette or hard disk so that you can use it as a QNX 4 filesystem, using *`fs-qnx4.so`* (p. 820). The default values are determined from the current configuration of the specified drive.

We recommend that you use `dinit` to initialize the QNX 4 filesystem, and *`dloader`* (p. 633) to make it bootable. The `dinit` bootloader options are for backwards compatibility reasons, but aren't generally used anymore.

If the disk is a hard disk, you need to specify the -h or -H (hard) option. This option helps protect you against typing errors that might cause `dinit` to initialize your hard disk. To initialize a hard disk, you must be the superuser.

Copyright © 2014, QNX Software Systems Limited

After initializing a hard disk with `dinit`, you should use the *dcheck* (p. 175) utility to remove any bad blocks from the disk allocation bitmap. For example:

```
dinit -h /dev/hd0t77
dcheck -m /dev/hd0t77
```

When `dinit` initializes a disk, it writes a loader in the first block. If the disk is a floppy diskette, the loader is the bootstrap loader, else it's the secondary (or partition) loader. If you need to rewrite the loader without reinitializing the disk, specify the -b option.

The -m option lets you change the message the OS displays when booting from disk. Normally, the message is:

```
Press ESC to boot alternate OS.
```

Your new message may contain up to 30 characters plus the trailing period. You can specify the minimum message of "`.`" by specifying -m. for the option.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

Copyright © 2014, QNX Software Systems Limited

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

Initialize a hard disk:

```
dinit -h /dev/hd0t77
```

Initialize a floppy disk:

```
dinit /dev/fd0
```

Pause before initializing hard disk:

```
dinit -hp /dev/hd0t77
```

Exit status:

0

Successful.

>0

An error occurred.

Caveats:

Don't use the -r option unless you know exactly what you're doing. You use the -r option only after a disaster has destroyed the first few blocks of your disk (e.g. a power failure occurred while the disk was being updated). In order for any damage to be repaired, you must follow `dinit -r` with this command:

```
chkfsys mountpoint
```


Copyright © 2014, QNX Software Systems Limited

dirname

Return the directory portion of a pathname (POSIX)

Syntax:

```
dirname string
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

None.

Description:

The `dirname` utility returns a portion of the *string* operand to standard output. The *string* operand represents a valid pathname whose format is:

directory_pathname/base_filename

The `dirname` utility writes the *directory_pathname* component to standard output.

If *string* is `//`, then `//` is returned. Any other string consisting entirely of slash characters causes a single slash to be returned.

You'll use the `dirname` utility most often within shell scripts, where it's normally invoked inside back-tick (`` `...` ``), or contained in `$(...)`.

Examples:

Command:	Output:
`dirname .`	`.`
`dirname ..`	`.`
`dirname ../.`	`..`
`dirname /usr/src/prog.c`	`/usr/src`
`dirname /usr/src/`	`/usr`
`dirname .../[fred]`	`...`

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

dloader

Write a boot loader to a disk

Syntax:

```
dloader [-v] [-d drive_number] [-F|H] [device loader] ...
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-d *drive_number*

Set the BIOS drive number for booting the second stage loader (`diskpc2`). This enables you to set up the loader to boot when the drive is configured as either a primary or secondary drive. Common values for *drive_number* are: `00` for the first floppy drive, `80` for the first hard drive, and `81` for the second hard drive.

If you don't specify this option or any of the other override options (-F, -H), a heuristic based on disk size and removability determines whether the drive is for a fixed or floppy disk. The order of precedence used to determine the drive number to be patched into the loader code is:

1. -d
2. -F
3. -H
4. heuristic

-F

Floppy loader override.

-H

Hard disk loader override.

-v

Be verbose.

Copyright © 2014, QNX Software Systems Limited

device

The name of the raw or partition disk mountpoint.

loader

The name of the loader.

Description:

The `dloader` utility writes a boot loader to a disk.

The `dloader` utility starts by looking for the loader you specified. If your loader name contains a `/` (slash), `dloader` assumes it is a complete pathname and looks for the loader using the path you gave; if not, `dloader` looks in the processor family subdirectory of ***QNX_TARGET*** (under QNX Neutrino, this would be ***$QNX_TARGET***`/x86/boot/sys`). To see the loaders provided in the default path, type `dloader` without any options.

The naming of QNX loaders provides a framework for specifying QNX and custom loaders. This framework consists of a loader name and a method of specifying optional variants.

Currently, the QNX Neutrino RTOS provides the following two standard loader names:

`pc1`

This is the standard first-stage (partition) loader on a PC.

`pc2`

This is the standard second-stage (QNX specific) loader on a PC.

Along with these two standard loaders, we also provide loader variants named `pc1-flop` and `pc2-flop`. These variants provide alternative loaders that are designed to work with floppy diskettes (or a hard disk with a capacity less than 8 Gbytes in an old PC with an old BIOS).

Similarly, users who wish to create and use their own loaders can specify them using unique variant names.

The device names for our loaders are prefixed with `ipl-disk`. So if you look in `/x86/boot/sys/` under QNX Neutrino, you will see the following loaders:

- `ipl-diskpc1`
- `ipl-diskpc1-flop`
- `ipl-diskpc2`
- `ipl-diskpc2-flop`

If you write your own loaders, make sure you use this `ipl-disk` prefix.

Copyright © 2014, QNX Software Systems Limited

If the loader is in the `/x86/boot/sys/` path, you don't need to specify the `ipl-disk` prefix on the command line because `dloader` adds it for you. If your loader is in a different directory, you have to specify the exact path, including the prefix.

Assuming you specified a device and loader correctly, `dloader` opens the device in the path you specified and, if you selected the verbose option, displays its disk and partition information.

The specified loader data is then put together and written to the disk.

The floppy (-F) and hard disk (-H) override options let you force `dloader` to treat a fixed device (e.g. a hard disk) as if it were a removable device (e.g. a floppy), and vice-versa.

Examples:

To see a list of available disk loaders:

```
dloader
```

To write a PC partition loader to a hard disk:

```
dloader /dev/hd0 pc1
```

To write a custom partition loader to a hard disk:

```
dloader /dev/hd0 /home/joe/ipl-diskpc1-tst
```

To write a QNX-specific second-stage PC loader to a QNX partition:

```
dloader /dev/hd0t79 pc2
```

To write a QNX loader to a floppy disk :

```
dloader /dev/fd0 pc2-flop
```

Environment variables:

QNX_TARGET

The location of target backends on the host machine.

Exit status:

0

The loader was written to the disk.

Any other value

Copyright © 2014, QNX Software Systems Limited

An error occurred.

Errors:

When an error occurs, `dloader` sends a description of the error to *stderr*.

Copyright © 2014, QNX Software Systems Limited

dnssec-dsfromkey

DNSSEC Delegation Signer resource record generation tool

Syntax:

```
dnssec-dsfromkey [-v level] [-1] [-2] [-a alg] {keyfile}

dnssec-dsfromkey {-s} [-v level] [-1] [-2] [-a alg] [-c class]
 [-d dir]
                  {dnsname}
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-dsfromkey++NetBSD-current* in the NetBSD documentation.

Description:

The `dnssec-dsfromkey` utility outputs the Delegation Signer (DS) resource record (RR), as defined in RFC 3658 and RFC 4509, for the given key(s). For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-dsfromkey++NetBSD-current* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

dnssec-keyfromlabel

DNSSEC key generation tool

Syntax:

```
dnssec-keyfromlabel {-a algorithm} {-l label} [-c class] [-f
flag] [-k]
                    [-n nametype] [-p protocol] [-t type] [-v
 level] {name}
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-keyfromlabel++NetBSD-current* in the NetBSD documentation.

Description:

The `dnssec-keyfromlabel` utility gets keys with the given label from a crypto hardware and builds key files for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-keyfromlabel++NetBSD-current* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

dnssec-keygen

DNSSEC key-generation tool

Syntax:

```
dnssec-keygen {-a algorithm} {-b keysize} {-n nametype}
              [-c class] [-e] [-f flag] [-g generator]
              [-h] [-k] [-p protocol] [-r randomdev]
              [-s strength] [-t type] [-v level] {name}
```

Runs on:

QNX Neutrino

Options:

See
http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-keygen++NetBSD-5.0
in the NetBSD documentation.

Description:

The dnssec-keygen utility generates keys for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845. For more information, see
http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-keygen++NetBSD-5.0
in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

dnssec-signzone

DNSSEC zone-signing tool

Syntax:

```
dnssec-signzone [-a] [-c class] [-d directory] [-e end-time]
                [-f output-file] [-g] [-h] [-k key] [-l domain]

                 [-i interval] [-I input-format] [-j jitter]
                 [-N soa-serial-format] [-o origin]
                 [-O output-format] [-p] [-r randomdev]
                 [-s start-time] [-t] [-v level]
                 [-z] {zonefile} [key...]
```

Runs on:

QNX Neutrino

Options:

See
http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-signzone++NetBSD-5.0
in the NetBSD documentation.

Description:

The `dnssec-signzone` utility signs a zone. It generates NSEC and RRSIG records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a keyset file for each child zone. For more information, see
http://netbsd.gw.com/cgi-bin/man-cgi?dnssec-signzone++NetBSD-5.0
in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

dprepresize

Prepare a Power-Safe (`fs-qnx6.so`) filesystem for resizing

Syntax:

```
dprepresize [options] device
```

Runs on:

QNX Neutrino

Options:

-b *num*

Relocate allocations that are beyond this boundary to a lower allocation. The units are in logical blocks.

-d

A test mode that relocates all allocations to the lowest available allocation.

-t

A test mode that doesn't make changes to the media that would cause *chkqnx6fs* (p. 121) to fail.

-v *level*

Set the level of verbosity:

- 0: quiet mode (the default)
- 1: normal mode
- 2: debug output

device

The name of the file or device to operate against.

Description:

The `dprepresize` utility prepares a Power-Safe (*fs-qnx6.so* (p. 823)) filesystem for resizing later with *dresize* (p. 643) to fit potentially smaller media than was specified at creation time.

These tools let you use a large disk when you create the starting image and then load the image onto a smaller disk. When you load the image on a smaller device, the

Copyright © 2014, QNX Software Systems Limited

allocation tables will be too large, so you can then use the resizing tool to shrink the size of the bitmap and accompanying metadata, and update the superblocks with information that's valid for the smaller disk.

The `dprepresize` tool takes a command-line argument referred to as a *boundary*. Any allocated blocks that are above that boundary are relocated to something below it. This ensures that an image can fit on the smallest targeted device.

The first time you boot the system, you can run `dresize`. If the signature in the superblock indicates that the disk needs to be resized, the resizing operation starts. Once the size of the disk is known, the bitmap is resized, and the metadata is updated to reflect the new bitmap size. Both superblocks are then updated with the correct total blocks, free blocks, allocation groups, etc. Finally, a new header block is written with the correct superblock locations.

Examples:

Prepare a filesystem for resizing:

```
dprepresize -b811056 /dev/hd0t178
```

View the phases the tool is performing:

```
dprepresize -v1 -b811056 /dev/hd0t178
```

Should pass chkqnx6fs after running:

```
dprepresize -t -v1 -b811056 /dev/hd0t178
```

Test relocating all allocations:

```
dprepresize -d -t -v1 -b811056 /dev/hd0t178
```


Copyright © 2014, QNX Software Systems Limited

dresize

Resize a Power-Safe (`fs-qnx6.so`) filesystem

Syntax:

```
dresize [options] device
```

Runs on:

QNX Neutrino

Options:

-g num_groups

Specify the number of allocation groups.

-n num_inodes

Specify the number of inodes.

-v level

Set the level of verbosity:

- 0: quiet mode (the default)
- 1: normal mode
- 2: debug output

device

The name of the file or device to operate against.

Description:

The `dresize` utility resizes a Power-Safe (*`fs-qnx6.so`* (p. 823)) filesystem to fit potentially smaller media than was specified at creation time. You must have prepared the filesystem for this with *`dprepresize`* (p. 641).

These tools let you use a large disk when you create the starting image and then load the image onto a smaller disk. When you load the image on a smaller device, the allocation tables will be too large, so you can then use the resizing tool to shrink the size of the bitmap and accompanying metadata, and update the superblocks with information that's valid for the smaller disk.

Copyright © 2014, QNX Software Systems Limited

The first time you boot the system, you can run `dresize`. If the signature in the superblock indicates that the disk needs to be resized, the resizing operation starts. Once the size of the disk is known, the bitmap is resized, and the metadata is updated to reflect the new bitmap size. Both superblocks are then updated with the correct total blocks, free blocks, allocation groups, etc. Finally, a new header block is written with the correct superblock locations.

Examples:

Resize a filesystem with no output:

```
dresize /dev/hd0t178
```

Resize a filesystem with output:

```
dresize -v1 /dev/hd0t178
```

Resize a filesystem and format with 25000 inodes:

```
dresize -v1 -n25000 /dev/hd0t178
```

Resize a filesystem and format with 4 groups:

```
dresize -v1 -g4 /dev/hd0t178
```

Resize a filesystem with debug output:

```
dresize -v2 /dev/hd0t178
```


Copyright © 2014, QNX Software Systems Limited

ds

Data server that maintains a shared state among processes

Syntax:

```
ds &
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The data server is a process that maintains a shared state among other processes — it's like a global environment. Processes can store or retrieve data using a set of data server library calls. You can use it for many tasks, but specifically you can access it from the Slinger webserver in support of dynamic HTML. The HTTP server *slinger* (p. 1780) makes use of the data server and the data server library.

Data is stored using variable names that represent buffers of data. All of the variables are global, *any* process can access them (no attempt is made to restrict access), so only one instance of the variable (*name*) can exist in the data server.

Data server library

The data server library consists of these functions, which are described in the *C Library Reference*:

ds_register()

Register your application with the data server.

ds_deregister()

Deregister your application with the data server.

ds_create()

Create a data server variable.

ds_clear()

Delete a data server variable.

Copyright © 2014, QNX Software Systems Limited

ds_set()

Set a data server variable.

ds_get()

Get a data server variable.

ds_flags()

Set the flags for a data server variable.

Examples:

Here's a simple (and nonfunctional) example of monitoring the temperature of an oven from a remote client:

This example uses HTML as the interface to the data server, but the data server isn't limited to HTML.

Here's what the `qnxvar` tokens on the client's web page look like:

```
<!--Show the current oven temperature-->
<!--#qnxvar format="<P>The oven temperature is %s degrees F." -->
<!--#qnxvar read="oven1 5" -->
```

If the temperature of the oven is currently 500 degrees F, the output looks like:

```
<P>The oven temperature is 500 degrees F.
```

Here's what the application monitoring the oven looks like:

```
// This program obtains the temperature of an oven, and
// then updates a data variable in the data server, to
// be read by slinger if the appropriate token is in an
// html page slinger is serving.

#include <stdlib.h>
#include <stdio.h>
#include <ds.h>
#include <string.h>

#define MAXLEN 4

int main(void)
{
  ds_t ds_descriptor;
  char ovenID[7], oven_temp[MAXLEN], flag=0;
  int length = MAXLEN;

  ds_descriptor = ds_register();
  if(ds_descriptor==-1){
    perror("ds_register");
    exit(1);
  }

  strcpy(ovenID,"oven1");

  if(ds_create(ds_descriptor, ovenID, flag, 0)==-1){
    perror("ds_create");
    exit(1);
  }

  // Obtain the an initial temperature for the oven
```


Copyright © 2014, QNX Software Systems Limited

```
    // to initialize the data server variable. strcpy
    // that value into oven_temp

    ds_set(ds_descriptor,ovenID,oven_temp,length);

    //Now let's update the temperature at some time interval

    while(1)
    //you might want some kind of decision to exit the program.
    {

       //obtain the current temperature from the oven

       //strcpy that temp reading into the oven_temp variable

       ds_set(ds_descriptor,ovenID,oven_temp,length);

       //wait a predetermined amount of time

     }

     ds_clear(ds_descriptor,ovenID);
     ds_deregister(ds_descriptor);

}
```

The output HTML page reflects the current temperature stored in the data server.

When this process exits, the data server variable is no longer available because the *flag* argument passed to *ds_create()* was 0.

If this application needs some data passed from HTML text, another variable is created by calling *ds_create()* and is used to pass information to it by using the `qnxvar write` token in some HTML text. The application gets the data by calling *ds_get()* and/or react to the change in data by receiving a proxy or signal.

Here's what's happening:

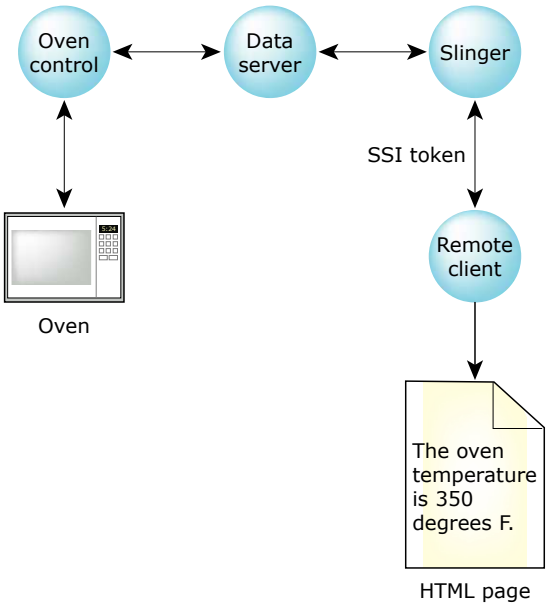


Figure 7: Monitoring an oven via the data server.

In summary:

Copyright © 2014, QNX Software Systems Limited

- Use the *qnxvar read* (p. 1787) token to display a value on the HTML page and use *ds_get()* to get the data server variable.
- Use the *qnxvar write* (p. 1787) token to change a variable on the data server process and use *ds_set()* to set the data server variable.

For information about the `qnxvar` token, see *slinger* (p. 1780).

Copyright © 2014, QNX Software Systems Limited

du

Estimate disk space usage (POSIX)

Syntax:

```
du [-a|-s] [-kpqx] [file...]
```

Runs on:

QNX Neutrino

Options:

-a

Generate a report for each file in the directory tree. If you don't specify this option, du makes a report for each directory only. The report shows the total space allocated to all files under the directory, including the directory itself.

-k

Report the space figures in Kbytes (the default is 512-byte blocks).

-p

Report the space figures in bytes (the default is 512-byte blocks). Also, ensure that du generates error messages when it can't process existing files (unless the -q option is specified).

-q

Be quiet; suppress error messages when du can't provide statistics on files, or can't read directories.

-s

Give the total figures for each of the specified files, rather than the totals for any subdirectories.

-x

Don't span device boundaries (used to determine how much space on a particular device is consumed by a directory tree).

file

Copyright © 2014, QNX Software Systems Limited

The pathname of a file whose size is to be displayed. If you don't specify any files, the current directory is used; du behaves as if the filename dot (.) were given.

Description:

The du utility prints the amount of file space allocated to the specified files. If you name a directory, all files in that directory are reported; subdirectories are traversed recursively. If a file has multiple links, the space allocated to that file is counted only once.

The space figures are displayed in 512-byte blocks by default. If you want du to print the size in bytes, specify the -p option. The sizes output by du when you specify the -p option are accurate with one exception: the numbers may be slightly higher than expected because they include extent blocks that are part of filesystem overhead associated with the file, but don't contain actual data.

If you specify nondirectories, they aren't listed unless you specify the -a option.

All results are written to the standard output. Errors may result in diagnostic messages to the standard error. Standard input isn't used.

Examples:

Estimate disk space consumed by the contents of /tmp, in kbytes:

```
du -k /tmp
```

Estimate the total space occupied by the contents of the current directory:

```
du -s
```

Exit status:

0

Successful completion.

>0

An error occurred. This doesn't include failure to read files or directories.

Copyright © 2014, QNX Software Systems Limited

dumpefs

Dump an embedded filesystem

Syntax:

```
dumpefs [-tuv]  embedded files
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-t

Dump the “text” of each extent.

-u

Output a unit summary, including the size of the filesystem, and the amount and percentage of space used.

-v

Be verbose.

Description:

The `dumpefs` utility dumps the contents of an embedded filesystem.

Copyright © 2014, QNX Software Systems Limited

dumper

Dump the postmortem state of a program (QNX)

Syntax:

```
dumper [-bFfmnPStvw] [-D path] [-d path] [-N max_files]
       [-p pid] [-s size[G|M|K]]
       [-U user_name | uid[:gid[,sup_gid]*]]]
       [-z level] &
```

Runs on:

QNX Neutrino

Options:

-b

(QNX Neutrino 6.6 or later) Attempt to slog a backtrace (`libbacktrace.so.1` must be available).

-D *path*

(QNX Neutrino 6.6 or later) The same as -d, but without querying `authman`.

-d *path*

The directory in which to place dumps, if `authman` doesn't supply an application sandbox path. The default is the home directory of user that started the process, or `/tmp` if none.

-F

(QNX Neutrino 6.6 or later) Run at a fixed priority.

-f

(QNX Neutrino 6.6 or later) Follow soft links for the creation of the dump files. The use of this option has security implications.

-m

Don't dump memory.

-N *max_files*

(QNX Neutrino 6.6 or later) Save sequential dumps, to a maximum of the given number of files. Each dump is saved in a file whose name is in the form:

Copyright © 2014, QNX Software Systems Limited

executable.num.core

where *num* starts at 1 and increases until the filename doesn't already exist.

-n

Save sequential dumps. Each dump is saved in a file whose name is in the form:

executable.num.core

where *num* starts at 1 and increases until the filename doesn't already exist.

-S

(QNX Neutrino 6.6 or later) Disable the dumping of shared memory mappings.

-P

Dump the physical memory mappings.

-p ***pid***

Save a dump file for this process immediately, and then exit dumper.

-s ***size*****[GIMIK]**

Set the maximum core size, in bytes.

-t

Dump the stack of the errant thread only, instead of for all threads.

-U ***user_name***

-U ***uid*****[:***gid***[,***sup_gid***]*]]**

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as root:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the /etc/passwd and /etc/group files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

-v

Be verbose.

-w

Make core files world-readable.

Copyright © 2014, QNX Software Systems Limited

-z level

Use *gzip* (p. 921) to compress the core files. The compression level must be in the range from 1 (fastest) through 9 (best compressed).

Description:

The `dumper` utility runs in the background and provides a postmortem dump service for all processes. Whenever a program terminates abnormally, a dump of the current state of the program is written to disk. The dump filename is the same as the program name with a `.core` extension. For example, if the program name is `experiment`, the dump is written to `experiment.core` in your home directory.

- On a QNX Neutrino system, `dumper` starts with `dumper -d /var/dumps`. You can use the -d option to force all dumps into a directory other than `/var/dumps`.
- Dump files can be large, so make sure the destination filesystem has lots of space.

The -p option lets you get a dump immediately for a particular process. If you specify -p, `dumper` doesn't run in the background, but exits right away.

You can use a debugger such as *gdb* (p. 892) to examine a dump file:

```
gdb program_binary program_core
```

A program may terminate in one of two ways: it may exit cleanly under its own control, returning an exit status, or it may be forcibly terminated by the receipt of a signal that it isn't prepared to handle. In the latter case, `dumper` writes a dump file for the following set of signals:

Signal	Description
`SIGABRT`	Program-called abort function
`SIGBUS`	Parity error
`SIGEMT`	EMT instruction (emulation trap) Note that `SIGEMT` and `SIGDEADLK` refer to the same signal.
`SIGFPE`	Floating-point error or division by zero
`SIGILL`	Illegal instruction executed
`SIGQUIT`	Quit
`SIGSEGV`	Segmentation violation

Copyright © 2014, QNX Software Systems Limited

Signal	Description
SIGSYS	Bad argument to a system call
SIGTRAP	Trace trap (not reset when caught)
SIGXCPU	Exceeded the CPU limit
SIGXFSZ	Exceeded the file size limit

You can force the dump of a running program by setting one of the preceding signals, assuming that the program isn't masking or handling the signal itself.

For example, to force a dump using the *kill* (p. 1026) command and a process ID (*pid*):

```
kill -SIGABRT pid
```

To force a dump using the *slay* (p. 1774) utility and the process name:

```
slay -s SIGABRT process_name
```

Examples:

Start `dumper`, with dump files to be written to the default directory:

```
dumper &
```

Start `dumper`, with dump files to be placed in the directory `/home/dumps`:

```
dumper -d /home/dumps &
```

Register for dump notifications:

```
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/dcmd_dumper.h>
#include <fcntl.h>
#include <unistd.h>
#include <devctl.h>
#include <sys/neutrino.h>

int
dumper_notify_attach(struct sigevent *devent)
{
    int dumper_fd;
    dumper_fd = open("/proc/dumper",O_RDONLY);
    if (dumper_fd >= 0) {
        devctl(dumper_fd, DCMD_DUMPER_NOTIFYEVENT, devent, sizeof(*devent), NULL);
        fcntl(dumper_fd, F_SETFD, FD_CLOEXEC);
    } else {
        dumper_fd = -1;
    }
    return dumper_fd;
}

#define DUMP_PULSE_CODE 0x50

int
main(int argc, const char *argv[], const char*envp[]){
    int dp_chid=-1;
```

Copyright © 2014, QNX Software Systems Limited

```
    int dp_coid=-1;
    struct sigevent devent;
    struct _pulse gpulse;
    int dumper_fd=-1;
    int rcvid;
    pid_t pid;

    // create death pulses channel
    dp_chid = ChannelCreate(_NTO_CHF_FIXED_PRIORITY);
    if(dp_chid==-1){
        perror("ERROR: ChannelCreate");
        exit( -1 );
    }
    dp_coid = ConnectAttach(0, 0, dp_chid, _NTO_SIDE_CHANNEL, _NTO_COF_CLOEXEC);
    if(dp_coid==-1){
        perror("ERROR: ConnectAttach");
        exit( -1 );
    }
    SIGEV_PULSE_INIT(&devent, dp_coid, sched_get_priority_max(SCHED_RR), DUMP_PULSE_CODE, -1);
    dumper_fd=dumper_notify_attach(&devent);
    if(dumper_fd==-1){
        perror("ERROR: opening /proc/dumper");
        exit( -1 );
    }
    for (;;) {
        // Blocks waiting for a pulse
        rcvid = MsgReceivePulse(dp_chid, &gpulse, sizeof(gpulse),NULL);
        switch (gpulse.code) {
            case DUMP_PULSE_CODE: // something died
                pid = gpulse.value.sival_int;
                fprintf(stderr, "Received Death Pulse code %"PRId8"\n" , gpulse.code);
                fprintf(stderr, "Process Pid %d died abnormally\n" , pid);
                break;
            default:
                fprintf(stderr, "Unknown pulse code: %"PRId8"\n" , gpulse.code);
                break;
        }
    }
    if (dumper_fd >=0)
        close(dumper_fd);
    if (dp_coid >=0)
        ConnectDetach(dp_coid);
    if (dp_chid >=0)
        ChannelDestroy(dp_chid);
    exit(0);
}
```

Files:

/proc/dumper

A special entry in the `/proc` filesystem (see *procnto** (p. 1586)) that receives notification when a process terminates abnormally.

Exit status:

The `dumper` utility normally doesn't terminate. However, it may terminate if it encounters an error on startup (for instance, if it wasn't run by `root`) or if it receives a signal.

0

A signal was received and dumper shut down successfully.

1

Copyright © 2014, QNX Software Systems Limited

An error was encountered on startup (not run by `root` or bad command-line options).

Copyright © 2014, QNX Software Systems Limited

dumpifs

Dump an image filesystem

Syntax:

```
dumpifs [-bmvx] [-d dir] [-f file] [-u file] image [files]
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-b

When using the -f or -x option, extract to the basenames of the files.

-d *dir*

The directory to which to extract files. The default is the current working directory.

-f *file*

Extract the named file.

-m

Display the MD5 checksum of each file in the image filesystem.

-u *file*

If the image is compressed, put an uncompressed copy in *file*.

-v

Verbose operation. Specify more than one `v` to display more information.

-x

Extract the *files* specified after the *image*.

Description:

The `dumpifs` utility dumps the contents of an image filesystem.

You can also use `dumpifs` with the -f or -x option to extract files from the image filesystem. The files are extracted into the current working directory, or into the directory you specify with the -d option. If you specify -b, the files are put directly into the

Copyright © 2014, QNX Software Systems Limited

directory; if you don't specify -b, the files are extracted to the pathname specified in the image.

Examples:

```
$ dumpifs shell.ifs
   Offset     Size  Name
        0      288  *.boot
      288      100  Startup-header flags1=0x1 paddr_bias=0
      388     6008  startup.*
     6390       59  Image-header mountpoint=/
     63ec      1ac  Image-directory
     ----     ----  Root-dirent
     6598       8c  proc/boot/data1
     6624       5c  proc/boot/.script
     6680       14  proc/boot/data2
     7000    2c02c  proc/boot/procnto
    34000    12ad0  proc/boot/devc-con
    47000     b66c  proc/boot/esh
    53000     d7fc  proc/boot/ls
    61000     7394  proc/boot/cat
Checksums: image=0x6d5fb484 startup=0x274d7c89
```

Caveats:

This utility will not work on an image that has been built using a filter such as `srec` (for more information on image filters, see *mkifs* (p. 1264)). If you wanted to run `dumpifs` on such an image, you would build the image omitting the filter stage in your `mkifs` buildfile (you would then need to run the filter by hand later in order to make a viable image for your target).

Copyright © 2014, QNX Software Systems Limited

dvfs_client

Client for interacting with a Dynamic Voltage Frequency Scaling driver

Syntax:

```
dvfs_client
```

Runs on:

QNX Neutrino

Targets:

ARMv7

Options:

None.

Description:

The `dvfs_client` utility provides a way for you to interact with a *`dvfsmgr-*`* (p. 663) driver.

Run this utility with no options. It will register itself under `/dev/dvfs_client_#`, where # is the instance number of the client. You can have a maximum of nine concurrent DVFS clients.

Use *`echo`* (p. 668) and *`cat`* (p. 95) to send commands to the DVFS driver. For example, to read the driver's status:

```
# cat /dev/dvfs_client_1
```

You can set DVFS driver's mode like this:

```
echo option > /dev/dvfs_client_#
```

You can do this only if the driver is in automatic mode.

The *option* is one of:

- MM: set to manual mode.
- MS: set to semi-automatic mode.
- MA: set to automatic mode.

Copyright © 2014, QNX Software Systems Limited

Change the power level like this:

```
echo level > /dev/dvfs_client_#
```

This can be done only by the client that sets the DVFS driver's mode to manual or semi-automatic.

The *level* is one of:

- PU: increase the power level.
- PD: decrease the power level.
- PX: set the power level to the maximum.
- PM: set the power level to the minimum.

To set minimum level of performance:

```
echo Slevel > /dev/dvfs_client_#
```

To unset a previously set minimum level:

```
echo U > /dev/dvfs_client_#
```

You can set or unset the minimum level only if the DVFS driver is in automatic mode.

The DVFS driver responds to the device control (*devctl()*) messages listed below. Some commands may be valid only for a certain driver mode, as indicated.

DVFS_DEVCTL_SET_MIN

Set the minimum level of power (automatic mode).

DVFS_DEVCTL_UNSET_MIN

Unset the minimum power level (automatic mode).

DVFS_DEVCTL_SET_MODE

Set the driver mode.

DVFS_DEVCTL_SET_PWR_LVL

Set the driver's power level (manual or semi-automatic).

DVFS_DEVCTL_GETSTATUS

Get the driver's status.

DVFS_DEVCTL_RUN_MAX

Run at the maximum power level (manual or semi-automatic).

DVFS_DEVCTL_RUN_MIN

Copyright © 2014, QNX Software Systems Limited

Run at the minimum power level (manual or semi-automatic).

Copyright © 2014, QNX Software Systems Limited

dvfsmgr-*

Dynamic Voltage Frequency Scaling driver

Syntax:

```
dvfsmgr [vwp:i:c:]
```

Runs on:

QNX Neutrino

Options:

The -c and -v options can be used by all users. The other options should be used only by advanced users (with guidance from QNX).

c ***cfg_file***

The path to the DVFS configuration file. The default is `/etc/system/config/dvfs.conf`.

-i ***interval***

The CPU accounting interval, in milliseconds (default: 1000ms).

-p ***prio***

The CPU accounting priority (default:51).

-v

Be verbose. Additional v options cause more verbosity.

-w

Enable the wfi workaround (default: disabled).

Description:

The `dvfsmgr-*` driver manages Dynamic Voltage Frequency Scaling (DVFS) for a system on a chip (Soc). DVFS allows SoCs to run at various power levels based on CPU load and/or temperature. This system-level power management could be beneficial in saving power during low CPU loads and ensuring that the chip's thermal limits are never reached.

This driver provides some level of power and thermal management from a system level. The driver is responsible for the following tasks:

Copyright © 2014, QNX Software Systems Limited

- monitoring the CPU load (MPU load)
- monitoring the SoC temperature (if possible)
- scaling the voltage and the frequency of MPU, based on the reported CPU load and/or SoC temperature
- providing an API for applications to send commands or read various information

Driver-level power management, CPU core scaling, thread migration, non-MPU voltage/frequency monitoring/scaling and off-chip cooling device control are out of scope of this driver.

The *dvfs_client* (p. 660) utility provides a way for you to interact with the DVFS driver.

DVFS configuration file

You need to provide a DVFS configuration file to the driver. This file has the following columns, separated by either spaces or tabs:

Level number

The level number (0-based) of each row within the configuration file.

Up threshold

The “UP” or higher power-transition threshold. If the CPU load exceeds the value for a given level, a transition occurs (if allowed).

Down threshold

The “DOWN” or lower power transition threshold. If the CPU load drops below this value for a given level, a transition occurs.

Up temperature threshold

The temperature (in degrees Celsius) at which a certain power level is disabled and a transition to a lower level occurs (if allowed).

Down temperature threshold

The temperature (in degree Celsius) at which a certain power level is reenabled. A transition may or may not occur (if allowed, and depending on CPU load).

The DVFS driver parses the configuration file on startup and extracts all relevant information from it. A line starting with ‘#’ is considered a comment. The driver reports any errors encountered while parsing through this file. A DVFS configuration file with 3 levels of transition is shown below:

```
#DVFS Config file
# Level   UP_threshold   DOWN_threshold    Up_temperature   Down_temperature
    0        100            40                80               78
```


Copyright © 2014, QNX Software Systems Limited

```
1         80            20              90              88
2         60             0              95              93
```

A sample configuration file is included with the driver. You can modify this file to meet your specific needs.

Don't change the number of levels in the configuration file. This file gets mapped to an internal power table that's maintained by the driver, and adding or removing levels might result in unexpected behavior. For example, if the provided sample configuration file contains five levels, the modified file must also contain five levels.

Operating modes

The DVFS driver supports the following operating modes, to ensure maximum flexibility:

Automatic mode

This is the default mode of the driver. The driver automatically adjusts voltage and frequency, based on the CPU load and temperature. Multiple clients can obtain the current status of the driver, which contains information such as load, temperature, mode of operation and so on. In this mode, clients can't change the power levels, and such requests are ignored.

This mode, however, allows an application to request a minimum level of operation, below which the driver shouldn't operate (unless forced by high temperatures). The driver is in charge of notifying the clients if any change to their desired setting is made.

Semi-automatic mode

Semi-automatic mode can be activated by only one client at any given time.

The driver may be changed to this mode when the appropriate request is made from a client. When a client successfully changes the mode to semi-automatic, it's in charge of requesting power level change, based on the status of the driver. No other client is allowed to change the mode and/or power level of the driver. The driver decreases power levels due only to temperature restrictions.

The controlling client may relinquish control by setting the mode of the driver back to automatic mode. Once the driver is back in automatic mode, any client may change the mode to manual or semi-automatic.

Manual mode

Manual mode can be activated by only one client at any given time. This mode is similar to semi-automatic, with the exception that the driver doesn't modify power levels under any circumstances, and the controlling client has full control over the driver.

Copyright © 2014, QNX Software Systems Limited

Reading the driver status

To get the driver's status, read from `/dev/dvfs`. The DVFS driver returns the status of driver, along with all significant information regarding the driver. Here's an example:

```
# cat /dev/dvfs
----------------------------
   Core Temp = 95
   CPU<0> Load = 91
   CPU<1> Load = 68
   Mode = Auto (0)
----------------------------
  #  |  App  |  Thermal
----------------------------
  0  |   -   |    x
 *1  |   -   |    -
  2  |   -   |    -
----------------------------
```

The information that the driver returns depends on the platform.

In this example, the temperature of the core is 42°C, CPU loads for cores 0 and 1 are 91% and 68%, and the driver is running in automatic mode. Furthermore, three power levels are supported. Power level 0 (the highest level) is disabled by temperature, and the driver is running at power level 1 (shown with an asterisk).

Copyright © 2014, QNX Software Systems Limited

Chapter 6
E

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`enum-usb`	`usblauncher` — see the *Device Publishers Developer's Guide*

This chapter describes the utilities, etc. whose names start with “E”.

Copyright © 2014, QNX Software Systems Limited

echo

Write arguments to standard output (POSIX)

Syntax:

```
echo [-n] [string...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-n

Don't write a trailing newline character.

string

A string to be written to standard output.

Description:

The echo command is present both as a shell builtin (see the *echo* (p. 1062) command for *ksh* (p. 1029)) and as a standalone executable that can operate without the availability of the system shell. Both versions behave in a similar manner. To make sure you use the executable, specify the full path.

The echo utility writes its arguments, followed by a newline character, to standard output. If there are no arguments, only the newline character is written.

The echo utility supports the following escape sequences within *string*:

Escape	Description
\a	Write an alert character (the bell).
\b	Write a backspace character.
\c	Suppress the newline character that otherwise follows the final argument in the output. All characters following the \c in the arguments are ignored.
\f	Write a formfeed character.
\n	Write a newline character.

Copyright © 2014, QNX Software Systems Limited

Escape	Description
\r	Write a carriage-return character.
\t	Write a tab character.
\v	Write a vertical tab character.
\\	Write a backslash character.
\0*num*	Write an 8-bit value that's the ASCII character represented by the specified 1-, 2-, or 3-digit octal number *num*.

The escape sequences listed above are extensions to POSIX. For a more versatile utility that's portable, see *printf* (p. 1581).

Examples:

Echo the string `Hello, Mother\nHello, Father` to the standard output (note that `echo` appends a final trailing newline):

```
$ echo 'Hello, Mother\nHello, Father'
Hello, Mother
Hello, Father
$
```

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

ed

Text editor

Syntax:

```
ed [-] [-sx] [-p string] [file]
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-s

Suppress diagnostics. Use this option if the standard input to ed is from a script.

-x

Prompt for an encryption key that will be used for reads and writes.

-p *string*

Specify a command prompt.

Description:

The ed utility is a text editor.

Copyright © 2014, QNX Software Systems Limited

egrep

Extended regular expression `grep` (UNIX)

Syntax:

```
egrep [-cilnqsvx]
      [-e expression | -f expression_file]...
      [file...]
egrep [-cilnqsvx] expression [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

See *grep* (p. 914) for a complete listing.

Description:

The `egrep` utility does an extended regular expression `grep` (equivalent to `grep -E`). See the documentation for `grep` for details.

Exit status:

0

Lines were found matching the expression provided.

>0

An error occurred or no matching lines were found.

Copyright © 2014, QNX Software Systems Limited

elvis

Visual interface editor clone (UNIX)

Syntax:

```
elvis [options]... [+command] file...
```

Runs on:

QNX Neutrino

Options:

-c *command*

Begin editing by executing this *ex* (p. 698) command.

-e

Start up in colon (ex) command mode.

-f

Force nonregular files (directories or device special files) to be opened. By default, elvis refuses to open nonregular files.

-i

Start up in input mode.

-m [*file*]

Scan *file* for errors and position the cursor at the offending line number. If you don't specify *file*, errlist is used.

-R

Set read-only mode to prevent accidental overwriting of files.

-t *tag*

Start editing at the given tag. (See the *ctags* (p. 163) utility.)

-v

Start up in visual command mode.

+*command*

Begin editing by executing this *ex* (p. 698) command.

Copyright © 2014, QNX Software Systems Limited

file

A pathname of a file to be edited.

Description:

The `elvis` utility is an interactive fullscreen editor that is compatible with the Unix/POSIX `vi` editor. The `vi` utility in the QNX Neutrino RTOS is a link to `elvis`.

In `elvis`, changes are buffered and are written to the file only upon request. Since a temporary file is used for storage, `elvis` can edit files larger than the amount of memory available on the machine it's run on.

There are two major command modes:

- `vi`sual mode
- *ex* (p. 698) mode (also known as colon mode)

You can switch between modes. You'll probably use the visual command mode most of the time. This is the mode `elvis` normally starts up in.

In visual mode, the entire screen is filled with lines of text from your file (except the last screen line, which is reserved for status). You can view text and move around the file. Each keystroke is interpreted as part of a visual command. If you start typing text, it isn't inserted; instead, it's treated as part of a command. To insert text, you must first give an "insert text" command (see "Inserting text").

The *ex* (p. 698) mode is quite different. In this mode, `elvis` displays a "`:`" character on the bottom line of the screen as a prompt. You are then expected to type in a command and press **Enter**. The set of commands recognized in `ex` mode differs from those in visual mode, and are known as `ex` commands. A summary of these commands is found at the end of this description. (A line-oriented editor, `ex` is a predecessor of `vi`, and the two share common functionality.)

The capabilities of `elvis` are described throughout the following sections:

Copyright © 2014, QNX Software Systems Limited

Visual mode

Most visual mode commands are one keystroke long. The following sections list the operation performed by each keystroke, and any necessary options or operands.

Most commands may be preceded by a decimal number. Usually, this number specifies how many times the command is to be repeated. Without this number, the command in most cases executes just once.

Input mode

You can't enter text into your file directly from visual command mode. Instead, you must first give a command that puts you into input mode. The commands to do this are:

```
A, C, I, O, R, S,
a, c, i, o, s
```

Visual command mode looks a lot like text input mode. If you forget which mode you're in, just press **Esc**. If `elvis` beeps, you're in visual command mode. If `elvis` doesn't beep, you *were* in input mode — by pressing **Esc** you switch to visual command mode. One way or another, after you press **Esc**, `elvis` is ready for a command.

Copyright © 2014, QNX Software Systems Limited

Note that if the showmode option is set (this option described in the "Options" section below), the mode is displayed in the lower right-hand corner as either `Command` or `Input`. Also note that each mode uses a different cursor shape.

Operators

Among its various commands, `elvis` has seven basic but powerful operators that let you change, delete, cut, paste, shift, or filter text regions. Although they're described below in the appropriate sections, you should note that, for the most part, they all share a common form:

op<object>

Where *op* can be:

c

Change

d

Delete

y

Yank (copy)

>

Shift right

<

Shift left

!

Filter

One operator doesn't take an *object*:

p

Put

The *object* operand specifies the range of text that the *op* operator acts upon. The *object* can be any movement command or pattern search. (See "*Movement commands* (p. 677).") In this way, you can see the power of the operators. The `elvis` utility can act on words, sentences, regions up to a specified pattern — whatever range the *object* operand specifies.

Copyright © 2014, QNX Software Systems Limited

When you give a movement command as an operand, you can specify an optional count [*n*] to multiply the effect of the operand (e.g. `d2w` to delete next two words). When a count isn't provided, it typically defaults to 1.

Special cases

Operators are frequently used with lines. So, for simplicity, a few special forms of these commands operate only with line *objects*. The following list shows the syntax of these special forms. Note the count is optional, and precedes the operator (this differs from the above form):

[*n*]cc

Change *n* lines

[*n*]dd

Delete *n* lines

[*n*]Y or [*n*]yy

Yank *n* lines

[*n*]<<

Shift *n* lines to the left

[*n*]>>

Shift *n* lines to the right

Named buffers

When you yank, delete, change, shift, or filter text, `elvis` saves the affected region in a single unnamed cut buffer, which you can recall by the `p` (put) command. However, `elvis` also has 26 named buffers, `a` to `z`, that you can use to save blocks of text during an editing session. These buffers are often used when editing multiple files to move text around.

You can prefix the previously defined operator forms by `"a` through `"z` (represented below by "<a-z>) to indicate which cut buffer to use to store the modified text region:

"<a-z>*op*<*object*>

Normal cases.

"<a-z>[*n*]*op*

Special cases.

When you use the uppercase letters to denote the named buffers, the *objects* are appended to the buffer:

Copyright © 2014, QNX Software Systems Limited

"<A-Z>*op*<*object*>

Normal cases.

"<A-Z>[*n*]*op*

Special cases.

Movement commands

The following movement commands provide a convenient means to position the cursor throughout the file being edited. You can precede most by an optional count to repeat the action. More importantly, you can use these movement commands as operands to the change, yank, delete, put commands to specify the range of action to be performed (see "Inserting text" and "Deleting, yanking, putting" sections).

The movement commands operate on the following text objects:

Character

A single character.

Sentence

A sequence of text ending in one of the following characters, followed by two spaces or a newline:

```
. ! ?
```

Paragraph

A paragraph starts after an empty line or any of the pairs of characters defined with the `:set pa=` option are found.

Section

A section starts where any of the pairs of characters defined with the `:set se=` option are found.

Cursor movement

To move the cursor, you can use the keypad arrow keys; you can also use the **H**, **J**, **K**, and **L** keys.

[*n*]`j` or [*n*]down-arrow or [*n*]Ctrl–J or [*n*]Ctrl–N

Move down *n* lines (next).

[*n*]k or [*n*]up-arrow or [*n*]Ctrl–P

Move up *n* lines (previous).

[*n*]`l` or [*n*]right-arrow or [*n*]Space

Copyright © 2014, QNX Software Systems Limited

Move right *n* characters.

[*n*]h or [*n*]left-arrow or [*n*]Backspace

Move left *n* characters.

[*n*]-

Move to first nonblank character on *n*th line, in backward direction.

[*n*]+

Move to first nonblank character on *n*th line, in forward direction.

[*n*]$

Move to the end of the line.

^

Move to the beginning of the first word on the line.

0

Move to the left margin (first nonblank character) of current line.

[*n*]|

Move to the column specified by *n*.

[*n*]w

Move to the beginning of the next word.

[*n*]W

Move to the beginning of the next word that follows white space.

[*n*]b

Move to the previous word.

[*n*]B

Move to the previous word that's delimited by white space.

[*n*]e

Move to the end of the word.

[*n*]E

Move to end of the word that's delimited by white space.

Copyright © 2014, QNX Software Systems Limited

[*n*]G

Move to the specified line (default is last line of file for a single `G` command).

[*n*]f<*char*>

Move to the *n*th occurrence of *char* (in forward direction) on the current line. The cursor is placed at the matched character.

[*n*]F<*char*>

Move to the *n*th occurrence of char (in backward direction) on the current line. The cursor is placed at the matched character.

[*n*]t<*char*>

Move to the *n*th occurrence of *char* (in forward direction) on the current line. The cursor is placed just before the matched character.

[*n*]T<*char*>

Move to the *n*th occurrence of char (in backward direction) on the current line. The cursor is placed just before the matched character.

[*n*];

Repeat previous `f`, `F`, `t`, or `T` command, in the same direction.

[*n*],

Repeat previous `f`, `F`, `t`, or `T` command, in the opposite search direction.

%

Move to matching parenthesis, bracket, or brace; i.e.: `( ) [ ] { }`

[*n*])

Move to the beginning of the next sentence.

[*n*](

Move to the beginning of the current sentence.

[*n*]}

Move to the beginning of the next paragraph.

[*n*]{

Move to the beginning of the current paragraph.

[*n*]]]

Copyright © 2014, QNX Software Systems Limited

Move to the beginning of the next section.

[*n*][[

Move to the beginning of the current section.

[*n*]H

Move to the top left position of screen. If *n* is specified, move the cursor to the beginning of the line *n* lines from the top of the screen.

[*n*]L

Move to the beginning of the last line on screen. If *n* is specified, move the cursor to the beginning of the line *n* lines from the bottom of the screen.

M

Move to the beginning of the middle line on the screen.

Marking

m<*a-z*>

Mark the current position with the character <*a-z*>.

'<*a-z*>

Move to the beginning of the marked line.

`<*a-z*>

Move to the exact position marked with the character <*a-z*>.

''

Move back to the beginning of the line where it was before the last "nonrelative" move.

``

Move back to the exact position it was before the last nonrelative move.

Tags

Tags can't be included as <*object*> operands for commands such as change, yank, put, delete, shift, and filter.

:ta *tag*

Edit the file containing *tag*. Position at *tag*.

Copyright © 2014, QNX Software Systems Limited

Ctrl–]

The word at the cursor position is taken as the tag, and the editor finds the word as with the `:ta` command.

Inserting text

The following commands enter input mode, where the text you enter (until you press **Esc**) is put into the file:

a

Append text after the current cursor position.

A

Append text at the end of current line.

i

Insert text before the current cursor position.

I

Insert text at beginning of current line.

o

Open a new line below line cursor is on, insert text there.

O

Open a new line above line cursor is on, insert text there.

c<*object*>

Change the text between the current position and the position specified by the <*object*> (movement operand or pattern). You can specify an optional count to multiply the effect of the <*object*> operand (e.g. `c2w` to change next two words).

If the range is within the current line, a `$` is displayed at the end of the <*object*> to indicate end-of-range. Otherwise, the text in the range is deleted and you're placed in input mode. When within a line, the specified range of text isn't deleted until you type **Enter**.

[*n*]cc

Change *n* lines.

R

Copyright © 2014, QNX Software Systems Limited

Replace the rest of the line with text. Rather than delete the text as with the change command, before going into insert mode, all text entered overwrites the current line until it has been completely replaced; at that point you're placed in input mode.

s

Substitute text for the current character (abbreviation for `cl`).

S

Substitute text for the current line (abbreviation for `cc`).

Input mode

In input mode, all keystrokes are inserted into the text at the cursor's position, except for the following:

Esc or Ctrl–[

Exit from input mode, back to command mode.

INTR

Interrupt execution (usually **Ctrl**–**C**, see *stty* (p. 1863)).

erase or Ctrl–H

Erase the character before the cursor.

Ctrl–W

Erase the last input word.

Ctrl–A

Insert a copy of the last input text.

Ctrl–D

Delete one indentation character.

Ctrl–L

Redraw the screen.

Ctrl–M or Enter

Insert a newline.

Ctrl–P

Insert the contents of the cut buffer.

Ctrl–R

Copyright © 2014, QNX Software Systems Limited

Redraw the screen (like **Ctrl**–**L**).

Ctrl–T

Insert an indent character.

Ctrl–U

Backspace to the beginning of the line.

Ctrl–V

Insert the following keystroke, even if special (e.g. **Ctrl**–**V Ctrl**–**L** inserts a form-feed).

Deleting, yanking, putting

As mentioned in the introduction, `elvis` supports very powerful operator constructs. In this section we describe the delete, yank, and put operators.

Delete

Delete the specified region and place the text into the unnamed buffer, or a named buffer if one is specified.

Yank

Make a copy of the specified region and place it into the unnamed buffer, or a named buffer if one is specified.

Put

Put the contents of the unnamed buffer, or of the named buffer if specified, into its new location.

When you have a sequence of text you want to copy or move to a different location, you first use the yank or delete operators to copy or delete the data into a buffer; you then use the `put` command to place the data at its new location.

If you're using named buffers, you can switch to another file before putting the text back. Thus you can copy from one file to another.

To copy the next four lines, yank them with a command such as `4yy` or `y4j`, move to a new location and put the text by typing the `p` command.

To delete the next four lines, type `4dd` or `d4j`.

To move the next four lines, use a command such as `4dd`or `d4j`, then move to a new location and put the text by typing `p` or `:pu`.

Filters

With the filter command, you can select regions of text and run them through any command and insert the output into the file. The text in the range specified from the

Copyright © 2014, QNX Software Systems Limited

current line to the delimited <*object*> is filtered through the command and replaces the region specified. The text that was in the region before being replaced by the output of the command is saved in the unnamed buffer, or in a named buffer if one was specified. If uppercase letters are used, `elvis` appends text to the named buffer.

!<*object*>*command*

Delete the specified text object into a buffer (unnamed, or named if given). Pass the specified text region to the standard input of the command, and replace it with the command's output. When you enter this command, the ! prompt doesn't appear until an <*object*> has been given.

Shifting text

The shift operators, < and >, shift all the lines delimited by the current line and the <*object*> operand. Text is shifted by the value of the shiftwidth option (see below). The forms of the shift command are:

><*object*>

Shift <*object*> to the right.

<<*object*>

Shift <*object*> to the left.

[*n*]>>

Shift next *n* lines to the right.

[*n*]<<

Shift next *n* lines to the left.

Named buffer prefixes don't work with the shift operator.

Miscellaneous commands

[*n*].

Repeat last text modifying command *n* times.

[*n*]r<*char*>

Replace current character with *char*.

D

Delete to the end of the line (abbreviation for d$)

Copyright © 2014, QNX Software Systems Limited

[*n*]J

Join the next line with the current line.

:[*x*,*y*]j

Join all the lines in the specified range.

[*n*]x

Delete *n* chars, to the right, including cursor position.

[*n*]X

Delete *n* chars to the left of the cursor.

[*n*]~

Reverse the case of the *n* next characters.

:[*x*,*y*]p

Display text in the specified range.

:[*x*,*y*]l

Display text in the specified range with tab and end-of-line markers.

:[*x*,*y*]nu

Display text in the specified range, with line numbers.

:so *file*

Read and execute the commands listed in *file*.

:ve

Display version number and compilation date of `elvis`.

Searching

A pattern used for searching and substituting is called a *regular expression*. While most characters match themselves in a search request, some have special meaning, as described in the table below. To be used in a search expression, these special characters must be preceded by a backslash (\).

Character	Meaning
^	Match “beginning of line”.
$	Match “end of line”.

Copyright © 2014, QNX Software Systems Limited

Character	Meaning
.	Match any single character except the newline.
\<	Match the beginning of a word.
\>	Match the end of a word.
[*string*]	Match any single character in *string*.
[^*string*]	Match any single character not in *string*.
[*x*-*y*]	Match any character between *x* and *y* (range).
[^*x*-*y*]	Match any character not between *x* and *y* (range).
*	Match any repeated characters.
\	Escape special characters.

The commands for searching are as follows:

/ [*pattern*] Enter

Search forward for *pattern*.

? [*pattern*] Enter

Search backward for *pattern*.

/ [*pattern*] /{+|-}*n*Enter

Go to the *n*th line relative to the line on which *pattern* is found, in forward direction.

? [*pattern*] ?{+|-}*n*Enter

Go to the *n*th line relative to the line on which *pattern* is found, in backward direction.

/Enter

Repeat the previous pattern search, in forward direction.

?Enter

Repeat the previous pattern search, in backward direction.

n

Repeat the previous pattern search.

Copyright © 2014, QNX Software Systems Limited

N

Repeat the previous pattern search, in reverse direction.

For example, to match `fred1`, `fred2`, or `fred3`:

```
/fred[1-3]
```

To match `Realtime OS` at the beginning of a line:

```
/^Realtime OS
```

To match `Realtime OS` at the end of a line:

```
/Realtime OS$
```

Global & substitute commands

The global and substitute commands can operate over a range of lines. You can specify a range wherever [*x*,*y*] is indicated. The first element, *x*, indicates the first line of the range, and the second element, *y*, indicates the last line. You can use line numbers or any of the following special characters as range elements:

.

The current line.

$

The last line in the file.

%

Same as 1,$ (i.e. the entire file).

n

Relative offset from current line.

A range element may also be a pattern specification:

```
/pattern/
```

or a marked location:

```
'<a-z>
```

For example, to print the lines from the next line containing `steve` until the first subsequent blank line:

```
/steve,/^$/!lp
```

Substitute command

Copyright © 2014, QNX Software Systems Limited

The substitute command substitutes text matching a pattern with replacement text:

```
:[x,y]s/pattern/replacement_text/[c][g][p]
```

If none of the modifiers c, g, or p is specified, this command replaces the first occurrence of the given pattern. You can modify this behavior by specifying any combination of the three modifiers:

c

Prompt before replacing.

g

Replace all matched occurrences on a line.

p

Display all lines containing the replaced text.

This command is very powerful when used in conjunction with the global command (see below).

Global command

The global command searches through the lines of the specified range — or through the whole file if no range ([*x,y*]) is specified — for lines that contain the pattern. The *command* is performed on each matching line. The global command is of the following form:

:[*x,y*]g/*pattern*/ *command*

Execute *command* for every line that matches *pattern*.

:[*x,y*]g!/*pattern*/ *command* or :[*x,y*]v/*pattern*/ *command*

Execute *command* for every line that doesn't match *pattern*.

You can combine the substitute and global commands, using the following syntax:

```
[x,y]g/pattern/s//replacement_text/
```

This command runs the substitute command on every occurrence of a matched pattern within the given range. The null pattern specification (i.e. `//`) indicates to the substitute command that it's to use the currently matched global pattern as the text it's to replace.

The following variations may also be used:

```
[x,y]g!/pattern/s//replacement_text/
[x,y]v/pattern/s//replacement_text/
```

For example, substitute the word `fred` with the word `barney` in lines 1 to 10:

```
1,10g/fred/s//barney
```


Copyright © 2014, QNX Software Systems Limited

Match every `mary` that's at the beginning of a line and prompt the user to confirm the substitution:

```
1,$g/^mary/s//dave/c
```

Undo and retrieving

On occasion, you need to undo the effects of a command:

u

Undo the effects of the last command that changed the edit buffer.

U

Undo the effects of all the text modifying commands performed on this line. Return the line to its original state.

The editor saves the last nine deleted blocks of text in save buffers. You can retrieve a buffer with the following commands:

"*n*p

Retrieve *n*th previous deletion (1-9). Place after the cursor.

"*n*P

Retrieve *n*th previous deletion (1-9). Place before the cursor.

If you accidentally select the wrong buffer, you can use the undo command to clear it and then try specifying a different buffer.

You can also use named buffers (see "*Deleting, yanking, putting* (p. 683)"):

"<a-z>p

Retrieve the contents of the named buffer. Place after the cursor.

"<a-z>P

Retrieve the contents of the named buffer. Place before the cursor.

Screen commands

The scrolling commands are as follows:

[*n*]Ctrl–D

Scroll down *n* lines (default is half page). *n* is remembered and becomes the default.

[*n*]Ctrl–U

Copyright © 2014, QNX Software Systems Limited

Scroll up *n* lines (default is half page). *n* is remembered and becomes the default.

[*n*]Pg Dn or [*n*]Ctrl–F

Jump forward *n* pages.

[*n*]Pg Up or [*n*]Ctrl–B

Jump backward *n* pages.

[*n*]Ctrl–E

Scroll down *n* lines.

[*n*]Ctrl–Y

Scroll up *n* lines.

To position the current line at different positions on the screen by scrolling forward/backward:

z.

Place line at center of screen.

z-

Place line at bottom of screen.

Ctrl–L or Ctrl–R

Redraw the screen.

The status commands are as follows:

Ctrl–G or :f

Display status line on bottom of screen.

Writing files

The following commands write to a file. You can precede all `w` commands by an [*x*,*y*] range. For example, specify `1,6w` to write the first 6 lines.

:w

Write changes to current file.

:w *file*

Write changes to *file*.

:w >>*file*

Copyright © 2014, QNX Software Systems Limited

Append changes to *file*.

:w! *file*

Force write to *file*.

:wq

Write the file and exit.

ZZ *OR* :x

Exit `elvis`. If any changes were made, the edit buffer is written to the file.

Editing other files

:e *file*

Edit *file*.

:e!

Re-edit current file, discarding unsaved changes.

:e! *file*

Edit *file*, discarding unsaved changes to the current file.

:e +*n file*

Edit *file*, start at line *n*.

:e # or Ctrl–^

Return to the previous position in the last edited file.

:n

Edit the next file in the argument list.

:n!

Edit the next file, discarding unsaved changes to the current one.

:n *args*

Use this new argument list.

:args

Show list of files, `[]` indicate current file.

:rew

Rewind list of files, re-edit first file.

Copyright © 2014, QNX Software Systems Limited

:rew!

Rewind the list of files, re-edit the first file, and discard any unsaved changes to the current file.

See also the :ta (tags) movement command.

Reading in a file

:r *file*

Read in *file* after the cursor.

:*n*r *file*

Read in *file* after line *n*.

:r !*cmd*

Read in the output of the named command.

Leaving elvis

ZZ *or* :x

Exit elvis. If any changes were made, the edit buffer is written to the file.

:wq

Write the file and exit.

:q

Exit from elvis.

:q!

Exit from elvis and discard any unsaved changes.

:Q

Escape to the ex line editor (to return, type elvis).

Escaping to a shell

To execute a single command from within elvis, you can use the following command:

:!*cmd*

Execute a single command, then return to elvis.

To execute more than one command, you can create a shell:

:sh

Copyright © 2014, QNX Software Systems Limited

Start a subshell. You can type **Ctrl**–**D** or `exit` to return to `elvis`.

Macros

Macros let you bind a single key to an arbitrary set of editing commands.

Defining a macro is simple: you define an *lhs* (left-hand side), which is the single character you want translated, followed by an *rhs* (right hand side), which is the sequence it is mapped into.

:map ***lhs rhs***

Create macro.

:unmap ***lhs***

Delete macro.

:map

Display macros.

For example, `:map q :wq`**Ctrl**–**VEnterEnter** maps `q` into `:wq`**Enter**. The **Ctrl**–**V** is needed to quote (escape) the first **Enter**, while the second **Enter** ends the `map` definition.

Abbreviations

Word abbreviations are similar to macros, but they expand a short word into a longer word or words. If you type the abbreviation as part of a longer word, it's left alone. Abbreviations are used in input mode primarily to save typing.

Substitution isn't performed until you type a nonalphanumeric character to mark the end of the word. If you type **Ctrl**–**V** before the nonalphanumeric character, `elvis` doesn't perform the substitution.

:ab ***abbr replacement_text***

Create an abbreviation.

:ab

Display abbreviations.

:una ***abbr***

Turn off this abbreviation.

For example:

```
:ab woof mary had a little ram
```

This inserts the text `mary had a little ram` whenever the word `woof` is typed.

Copyright © 2014, QNX Software Systems Limited

Options

You can set or examine options via the colon command `set`. The values of options affects the operation of subsequent commands.

There are three option types:

- those that toggle on or off; these are enabled by typing `:set` *option* and disabled by typing `:set no`*option*.
- those that expect a numeric value
- those that expect a string value; these are used by typing `:set` *option*=*value*

To print all the option settings, type:

```
:set all
```

For convenience, options have both a long descriptive name and a short name (shown in parentheses below) that's easier to type. You may use either interchangeably.

autoindent (ai)

Default: noai

In input mode, if autoindent is on, each added line begins with the same amount of leading white space as the line above it. Without autoindent, added lines begin at column zero.

autoprint (ap)

Default: ap

This option affects only `ex` mode. If this option is on, and either the cursor has moved to a different line or the previous command modified the file, `elvis` prints the current line.

autowrite (aw)

Default: noaw

If you've made modifications to the current file then try switching to another file, (e.g. `:tag`, or `:next`...), `elvis` normally prints an error message and refuses to switch. If this option is on, `elvis` writes the modified version of the current file and switches to the new file.

directory (dir)

Default: dir="/tmp"

`elvis` stores text in temporary files. This option lets you control which directory those temporary files appear in. The default is `/tmp`.

Copyright © 2014, QNX Software Systems Limited

You can set this option only in a `.exrc` file; once `elvis` is started, the directory can't be changed.

If your `/tmp` directory is on ramdisk, recovery isn't possible if the system is rebooted.

edcompatible (ed)

Default: noed

Remember `:s//` options.

errorbells (eb)

Default: eb

The `elvis` utility normally rings a bell when you do something wrong. This option lets you disable the bell.

ignorecase (ic)

Default: noic

Normally, when `elvis` searches for text, it distinguishes between uppercase and lowercase letters. When this option is on, uppercase and lowercase are seen as equivalent.

list (li)

Default: noli

In nolist mode (the default), `elvis` displays text in a "normal" manner — with tabs expanded to an appropriate number of spaces, etc. However, sometimes it's useful to have tab characters displayed differently. In list mode, each tab is displayed as `^I`, and a `$` is displayed at the end of each line.

magic (ma)

Default: ma

The search mechanism in `elvis` can accept *regular expressions* — strings in which certain characters have special meaning. The magic option is normally on, which causes these characters to be treated specially. If you turn the magic option off (`:set noma`), then all characters except ^ and $ are treated literally. Both ^ and $ retain their special meanings regardless of the setting of magic.

paragraphs (pa)

Copyright © 2014, QNX Software Systems Limited

Default: pa="PPppIPLPQP"

The { and } commands move the cursor forward or backward in increments of one paragraph. Paragraphs may be separated by blank lines or by a leading "dot" command of a text formatter. Different text formatters use different dot commands. This option lets you configure `elvis` to work with your text formatter.

It's assumed your formatter uses commands that start with a "." character at the front of a line, followed by a one- or two-character command name.

The value of the paragraphs option is a string in which each pair of characters is one possible form of your text formatter's paragraph command. For example, lines starting with `.IP`, `.LP`, or `.PP` are considered new paragraphs.

readonly (ro)

Default: noro

Prevent overwriting of original file. Normally, `elvis` lets you write back any file for which you have write permission. If you don't have write permission, all you can do is write the changed version of the file to a different file.

If you set the readonly option, `elvis` pretends you don't have write permission to any file you edit. This is useful when you mean to use `elvis` only to look at a file, not to change it. This way, you can't change the file accidentally.

This option is normally off, unless you use the `view` alias of `elvis`, which is like `elvis` except the readonly option is on.

report (re)

Default: re=5

Changes, deletes, and yanks may affect many lines. For commands affecting a lot of lines, `elvis` outputs a message saying what was done and how many lines were affected. This option lets you define what "a lot of lines" means. The default is 5, so a message is shown on the status line for any command affecting 5 or more lines.

scroll (sc)

Default: sc=12

Scroll amount for **Ctrl**–**U** and **Ctrl**–**D**. These commands normally scroll backward or forward by half a screenfull, but you can adjust this. The value of this option says how many lines those keys should scroll. You can also set this with [*n*]**Ctrl**–**D** and [*n*]**Ctrl**–**U**.

Copyright © 2014, QNX Software Systems Limited

sections (se)

Default: se="NHSHSSSEse"

The [[and]] commands move the cursor backward or forward in increments of one section. Sections may be delimited by a { character in column 1 (which is useful for C source code) or by means of a text formatter's "dot" commands.

This option lets you configure elvis to work with your text formatter's section command, in exactly the same way that the paragraphs option makes it work with the formatter's paragraphs command. For example, lines starting with .SH or .NH are considered new sections.

shell (sh)

Default: sh="/bin/sh"

When elvis forks a shell (perhaps for the :! or :sh commands), this is the program it uses as a shell. The normal default is /bin/sh. However, if you have set the ***SHELL*** environment variable, the default value is copied from the environment.

shiftwidth (sw)

Default: sw=8

The < and > commands shift text left or right by a uniform number of columns. The shiftwidth option defines that uniform number. The default is 8 spaces.

showmatch (sm)

Default: nosm

With showmatch set, every time you type), }, or] in input mode, elvis momentarily moves the cursor to the matching opening parenthesis or bracket.

showmode (smd)

Default: nosmd

In visual mode, it's easy to forget whether you're in the visual command mode, or input/replace mode. Normally, the showmode option is off, and you aren't informed as to which mode you're in. If you turn the showmode option on, a message appears in the lower right corner of your screen, telling you which mode you're in.

tabstop (ts)

Copyright © 2014, QNX Software Systems Limited

Default: ts=8

The width of tab characters.

term (te)

Default: te="$TERM"

This read-only option shows the name of the terminal entry that `elvis` is using for your terminal.

warn (wa)

Default: wa

If you've modified a file, but not yet written it back to disk, `elvis` normally prints a warning before executing a `:!`*cmd*" command. However, in nowarn mode, this warning isn't given.

Normally, `elvis` also prints a message after a successful search that wrapped at EOF. The [no]warn option also disables this warning.

wrapmargin (wm)

Default: wm=0

Wrap long lines in input mode. Normally (with wrapmargin=0), `elvis` lets you type in extremely long lines.

However, with wrapmargin set to something other than 0 (wrapmargin=70 is nice), long lines are automatically “wrapped” on a word break for lines longer than wrapmargin's setting.

wrapscan (ws)

Default: ws

Normally, when you search for something, `elvis` finds it no matter where it is in the file. `elvis` starts at the cursor position, and searches forward. If `elvis` reaches EOF without finding what you're looking for, it wraps around to continue searching from line 1, up to the current line.

If you turn off the wrapscan option (`:se nows`), and `elvis` subsequently reaches EOF during a search, it stops and says so.

You can configure the option settings with the ***EXINIT*** environment variable:

```
EXINIT="set ai aw"
export EXINIT
```

---

ex commands

Copyright © 2014, QNX Software Systems Limited

The following list of ex commands is intended for your convenience only. It's beyond the scope of this document to describe the operations of these commands in detail. Most of these commands have been indicated as alternatives to the visual mode commands described in the above sections.

The general form of these commands is:

```
:[x,y] command parameter
```

[Prefix]	Command	Short form
	abbrev *abbr text*	ab
[*line*]	append	a
	args	ar
[*x*[,*y*]]	change	c
[*x*[,*y*]]	copy *line*	co
[*x*[,*y*]]	delete ["*named_buffer*]	d
	edit[!] *file*	e
	file	f
[*x*[,*y*]]	global /*pattern*/*command*/	g
[*line*]	insert	i
[*x*[,*y*]]	join	j
[*x*[,*y*]]	list	l
	map *lhs rhs*	map
[*x*[,*y*]]	move *line*	m
	next[!]	n
[*x*[,*y*]]	number	nu
[*x*[,*y*]]	print	p
[*line*]	put ["*named_buffer*]	pu
	quit[!]	q
[*line*]	read *file* | !*command*	r
	rewind[!]	rew
	set [*option*]	se
	shell	sh

Copyright © 2014, QNX Software Systems Limited

[Prefix]	Command	Short form
	`source` *file*	`so`
[*x*[,*y*]]	`substitute` /*pattern*/*text*/[c][g][p]	`s`
	`tag`	`ta`
	`unabbrev` *abbr*	`una`
	`undo`	`u`
	`unmap` *lhs*	`unm`
	`version`	`ve`
	`visual`	`vi`
[*x*[,*y*]]	`write[!]`[[>>]*file*]	`w`
	`xit[!]`	`x`
[*x*[,*y*]]	`yank` ["*named_buffer*]	`ya`

The following `ex` commands have no long version:

Command	Form
`escape`	`!`*command*
`print next`	`CR`
`lshift`	`<`
`rshift`	`>`

See also:

Linda Lamb, *Learning the vi Editor*, O'Reilly and Associates, 1990

Files:

`/tmp/elv*`

During editing, text is stored in a temporary file in `/tmp`.

`tags`

The tag database created by *`ctags`* (p. 163) and used by the `:tags` command and the -t option.

***`$HOME`*`/.exrc`**

On startup, the contents of this file are executed as a series of `ex` commands.

Copyright © 2014, QNX Software Systems Limited

Environment variables:

LINES*, *COLUMNS

Overrides the screen size values associated with your terminal type.

EXINIT

Default `elvis` option settings. If set, the contents of this environment variable are executed on startup as a series of `ex` commands.

The `elvis` utility requires that the ***TERM*** environment variable be set to indicate your terminal type. For example, if `elvis` is running on a console, ***TERM*** should be set as follows:

```
export TERM=qansi
```

or

```
export TERM=qnx
```

depending on the mode your console is running in (QNX vs ANSI).

Contributing author:

Steve Kirkendall

Caveats:

When displayed, long lines scroll horizontally. On some implementations, these wrap onto multiple rows on the screen.

Under QNX Neutrino, you *must* have a `/tmp` directory for temporary files. To use a RAM disk as `/tmp`, try this:

```
ln -sP /dev/shmem /tmp
```

Copyright © 2014, QNX Software Systems Limited

env

Set environment for command invocation (POSIX)

Syntax:

```
env [-i] [name=value]... [command [arguments]]
```

Deprecated:

```
env - [name=value]... [command [arguments]]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-i

Ignore the environment that's otherwise inherited from the current shell. This restricts the environment for *command* to that specified by any *name=value* pairs.

-

(Deprecated) same as -i

name=value

Set the environment variable *name* to *value* and add it to the environment.

command

A command to be invoked.

Description:

The env utility obtains the current environment, modifies it according to its arguments, and executes *command* with the modified environment. If no command is specified, the resulting environment is displayed.

If you have removed the ***PATH*** environment variable from the environment, you must include the path to the command.

Examples:

Start a shell with only the ***SHELL*** and ***PATH*** environment variables:

```
env -i SHELL=/bin/sh PATH=/bin:/usr/bin sh
```


Copyright © 2014, QNX Software Systems Limited

Start a daemon process with no environment:

```
env -i /bin/cron
```

Exit status:

When a *command* is specified on the command line, the env utility attempts to *exec()* into that command. If the command is successfully launched, the exit status from env is that of the program it *exec()*ed into. Otherwise, env returns an exit status as follows:

0

(Assuming no command was specified) the resulting environment was successfully displayed.

>0

An error was present in the command-line parameters supplied or env was unable to launch the command specified on the command line.

Copyright © 2014, QNX Software Systems Limited

errno

Explain errno numbers (QNX Neutrino)

Syntax:

```
errno error_number...
```

Runs on:

QNX Neutrino

Options:

error_number

Error number to be explained.

Description:

The `errno` utility displays the string equivalent for the *error_numbers* supplied on the command line. The output is written to the standard output. If the command-line argument isn't a valid error number, a line is written to the standard error instead.

This utility is useful in cases where a program has indicated that a numerical error occurred and hasn't provided the ASCII string equivalent of that *errno* value. Users who don't have access to the C header file <`errno.h`> don't have the option of using:

```
grep (p. 914) error_number /usr/include/errno.h
```

The `errno` utility is more convenient and is available to all users.

Examples:

Find the string equivalent of error number 2:

```
$ errno 2
errno    2: ERROR ENOENT
$
```

Exit status:

>0

An error occurred (e.g. unknown *errno* value).

0

The error string was successfully written to the standard output.

Copyright © 2014, QNX Software Systems Limited

esh

Embedded shell (QNX Neutrino)

Syntax:

```
esh [-c command] [-irv] [script_file]
```

Runs on:

QNX Neutrino

Options:

-c *command*

Run this command.

-i

Enter interactive mode after running a script file. If you don't specify this option, `esh` terminates after running a script.

-r

Run in restricted mode. In this mode, you can't perform certain operations. If you try to, the error message, "Operation not permitted" is displayed.

Restricted operations include running executables that start with a slash, exporting variables, and reattaching standard input, output and error to another device. For more information, see the sections "*Command-line format* (p. 705)" and "*Builtin commands* (p. 707)."

-v

Be verbose: echo each command before executing it.

script_file

A file containing shell commands to execute.

Description:

The `esh` utility provides a subset of the functionality found in the standard shell, `/bin/sh`. You should find `esh` useful for situations where memory requirements are limited. For example, you could use it to run a simple system initialization file for an embedded system.

Command-line format

Copyright © 2014, QNX Software Systems Limited

In esh, command lines take this form:

```
command arg1 arg2 ... [redir-op file] [&]
```

Where:

command

The command to be executed. If it doesn't start with a slash, the command follows the path set by the ***PATH*** environment variable. In restricted mode, a command can't start with a slash.

redir-op file

A redirection operator. When a command is invoked, three standard files are set up in its environment. These files, *standard input*, *standard output*, and *standard error* output (*stdin*, *stdout*, *stderr*), are usually attached to the active terminal. You can redirect a command's standard input, standard output, and standard error as follows:

Specifying:	Will:
<*file*	Redirect standard input from this file.
>*file*	Redirect standard output to this file. If the file exists, it's overwritten; if the file doesn't exist, it's created.
>>*file*	Redirect standard output to this file. If the file exists, the information is appended to the end of the file; if the file doesn't exist, it's created.
2>*file*	Do the same as >*file*, but for standard error.
2>>*file*	Do the same as >>*file*, but for standard error.
&	If a command contains an unquoted &, then esh doesn't wait for the command to complete execution but immediately moves on to process the next command. The standard input of the command is redirected from /dev/null, and SIGINT and SIGQUIT are ignored.

Copyright © 2014, QNX Software Systems Limited

Filename expansion

You use most commands for manipulating files in one way or another. As such, esh has a filename "shorthand" (consisting of *, ?, [, and]) that you can use to specify the files that a command is to operate on. This shorthand is the same used by the standard shell. For more information, see "*Filename patterns* (p. 1048)" in ksh.

Quoting

The following characters have a special meaning in esh:

```
&  \  "  *  ?  [  space
```

To suppress the special meaning of these characters and keep their literal meaning, you use quoting.

To quote a sequence of characters or sequence of words, enclose the sequence in double quotes. To quote a single character, use double quotes or precede it with the escape character (\).

Escape character (backslash)

The escape character (\) preserves the literal meaning of the next character. You can't obtain a single backslash by quoting \ with double quotes. To obtain a backslash, enter \\ instead.

Double quotes

Enclosing characters and words in double quotes ("") preserves the literal meaning of all characters within double quotes, with the exception of the \ character. For example:

```
"ab cd"
```

represents a single, five-character argument.

You can keep the literal meaning of a double quote with the \ character. For example:

```
ab\"cd
```

represents the single, five-character argument ab"cd.

Builtin commands

The following commands are built into esh (that is, esh interprets and executes them internally):

- *. (dot)* (p. 708)
- *alias* (p. 708)
- *cd* (p. 708)
- *emount* (p. 708)
- *ewaitfor* (p. 709)
- *exec* (p. 710)

Copyright © 2014, QNX Software Systems Limited

- *exit* (p. 710)
- *export* (p. 710)
- *kill* (p. 710)
- *reopen* (p. 710)
- *set* (p. 710)
- *unset* (p. 711)

. (dot) command

```
. file
```

The . (dot) command reads and executes the commands from *file* within the current environment. The search path contained in the environment variable ***PATH*** is used to find the directory containing *file*. This command doesn't require that *file* be executable.

alias command

```
alias [name=value]...
```

Without arguments, `alias` lists all aliases and their values. If only the name is specified, its value is listed. Any name specified with a value defines an alias.

Alias expansion occurs when the first word of a statement is a defined alias, except when that alias is already being expanded.

List all aliases:

```
alias
```

Remove an alias:

```
alias name=
```

cd command

```
cd [directory]
```

Change the working directory of the current execution environment. If *directory* isn't specified, the value of the ***HOME*** environment variable becomes the new working directory.

emount command

```
emount special directory [fs_type]
```

Mount a special device. The arguments are:

special

The name of the special device.

mountpoint

Where to mount the device on your system.

Copyright © 2014, QNX Software Systems Limited

type

The type of filesystem or manager to mount:

type:	Filesystem or manager:
cd	*fs-cd.so* (p. 787)
cifs	*fs-cifs* (p. 790)
dos	*fs-dos.so* (p. 795)
ext2	*fs-ext2.so* (p. 807)
mac	*fs-mac.so* (p. 809)
nfs	*fs-nfs2* (p. 811), *fs-nfs3* (p. 814)
nt	*fs-nt.so* (p. 818)
qnx4	*fs-qnx4.so* (p. 820)
qnx6	*fs-qnx6.so* (p. 823)
udf	*fs-udf.so* (p. 829)

The default is qnx4.

The emount command was implemented in QNX Momentics 6.3.0 Service Pack 2.

ewaitfor command

```
ewaitfor path [max_seconds [delay]]
```

Wait until the given path exists. The arguments are:

path

The path to test.

max_seconds

The maximum number of seconds to wait for the file to appear. The default is 1 second.

delay

The number of milliseconds to wait between attempts. The default is 100 ms.

Copyright © 2014, QNX Software Systems Limited

The `ewaitfor` command was implemented in QNX Momentics 6.3.0 Service Pack 2.

`exec` command

```
exec [command [argument...]]
```

Execute a command and/or manipulate file descriptors.

The `exec` command opens, closes, or copies file descriptors as specified by any I/O redirections given as part of *argument*. If a command is specified, that command is spawned as a replacement for `esh`. Any specified arguments are passed to the spawned process.

If `esh` is operating in restricted mode (-r), you can't use the `exec` command to run a command whose path starts with a slash.

`exit` command

```
exit [n]
```

Cause `esh` to exit with the exit status specified by *n*. If *n* isn't specified, `esh` exits with the status of the last command executed.

`export` command

```
export name[=word]...
export -p
```

Mark environment variables for export. This makes them be in the environment of subsequently executed commands. If you specify the -p option, the names and values of all exported variables are written to the standard output.

If restricted mode (-r) is set, you can't use this command.

`kill` command

```
kill pid
```

Set a `SIGTERM` on the process with process ID *pid*.

`reopen` command

```
reopen [device]
```

Close standard input, standard output, and standard error and attach them to the specified device. This command is often used in startup scripts. If you don't specify a device, `/dev/con1` is used.

If restricted mode (-r) is set, you can't use this command.

`set` command

```
set -v
```


Copyright © 2014, QNX Software Systems Limited

Enable verbose mode; all commands are echoed to the standard output before they're executed.

unset command

```
unset variable
```

Unset the specified variable. If restricted mode (-r) is set, you can't use this command.

Examples:

Invoke a subshell:

```
esh
```

Invoke a subshell with a script file:

```
esh /etc/backup
```

Run the following command and exit:

```
esh -c "ls /bin"
```

Files:

/etc/esh

ASCII text file containing shell commands, executed when esh is started as a login shell.

Environment variables:

HOME

The pathname of the user's home directory.

LOGNAME

The user's *login* (p. 1121) name.

PATH

The directory search path used by esh for locating executable programs and esh shell scripts. To change ***PATH***, you must use the export command.

If ***PATH*** isn't in the existing environment when esh is invoked, it is set to /bin:/usr/bin. For more information on setting ***PATH***, see "Setting ***PATH*** and ***LD_LIBRARY_PATH***" in the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*.

SHELL

The pathname of the user's preferred shell.

Copyright © 2014, QNX Software Systems Limited

TERM

The terminal type.

TMPDIR

The name of a directory where utilities can create temporary files.

TZ

The timezone setting.

Caveats:

The current version of esh strips out single quotes (`'`), which means that many common uses of commands such as *find* (p. 750) fail.

Copyright © 2014, QNX Software Systems Limited

etfsctl

Control an embedded transaction filesystem

Syntax:

`etfsctl [`*`options`*`]`

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

-c

Make the filesystem on the device continue or resume operations.

-d *device*

Connect to the specified device:

`/dev/etfs1`

The raw partition for user extensions (such as boot images).

`/dev/etfs2`

The filesystem partition for `etfs` files.

-D

Request a defragmentation operation on the `.filetable` for the ETFS filesystem. The background processing of the ETFS filesystem does its own defragmenting as needed. This option lets you force it to happen on demand.

-e

Erase the device. For NAND flash, factory-marked bad blocks aren't erased. Blocks that become bad during normal use (worn-out blocks) are also skipped during the erasing.

If you wish, you can use the -l and -o options to specify the length and offset for the erasure. If used, these options must preceed the -e option.

Copyright © 2014, QNX Software Systems Limited

-f

Erase as in -e, then format an empty filesystem. Don't use this option with -w, since -w assumes an erased partition with no filesystem.

-i

Print info about the filesystem. See the "Description" below.

-l *len*

The length for the subsequent -e, -R, or -r option. The *len* is in bytes, but you can add a suffix of `K`, `M`, or `G`.

-o *offset*

The starting offset for the subsequent -e, -R, -r, -W, or -w option. The *offset* is in bytes, but you can add a suffix of `K`, `M`, or `G`.

-p

Operate in software-update mode.

-r *file*

Read all data from the device and save it in the specified file. The data is saved as a series of transactions. This data can be written to another flash part as long as that part has the same:

- cluster size
- block size (number of clusters in a block)

The data format is endian-neutral and free of any device-specific characteristics such as how it stores CRCs or ECCs. You can now read and write filesystems across different classes of devices, for example for NAND and RAM.

If you wish, you can use the -l and -o options to specify the length and offset for which to read. If used, these options must preceed the -r or -R option. If you specify the -l option, `etfsctl` doesn't read past this length.

-R *file*

Read all data from the device, including blank or erased blocks, and save it in the specified file. This is the raw version of -r option.

-s

Stop the filesystem on the device.

-S

Copyright © 2014, QNX Software Systems Limited

Similar to -s, but wait for the filesystem to stop before returning.

-w *file*

Write transactions from the specified file to the device. This transaction file can be created by reading it from this or another device via the -r option of `etfsctl` or by the *`mketfs`* (p. 1219) utility. The transactions are block-location-independent on the device. This allows bulk programming of devices with bad blocks in any location. The only requirement is that enough good blocks should be available to hold all transactions.

If you wish, you can use the -o option to specify the offset at which to write. If used, this option must precede the -w or -W option.

-W *file*

Write transactions from the specified file to the device. Also, copy any blank or erased blocks. This is a raw version of the -w option.

Description:

The `etfsctl` utility is used to manage an embedded transaction filesystem (ETFS). The utility interacts with the running filesystem using *devctl()* messages. Using `etfsctl`, you can erase and format a partition, read or write an entire transaction log (and thus its entire filesystem) from or to the device, stop the filesystem, make it continue or get statistical information.

Options are processed from left to right in order. The first option must be a -d device where:

`/dev/etfs1`

The raw partition for user extensions (such as boot images).

`/dev/etfs2`

The filesystem partition for `etfs` files.

The raw partition is used for user extensions, such as boot images, and is always at the start of the device. It may be zero bytes long if you don't need it. The filesystem partition consists of a series of transactions that together form a filesystem. You can use the -r option to read the transactions from the device and save them to a regular file, typically on another filesystem. You can then use the -w option to write this transaction log to another ETFS filesystem.

When writing, you must erase the filesystem first; failure to do so corrupts the data on the device.

Copyright © 2014, QNX Software Systems Limited

The -w option is most often used to write transaction logs created by the *mketfs* (p. 1219) utility.

You can request the filesystem to stop accepting new open requests by using the -s or -S option. Once the last file currently open by any application is closed, the filesystem enters the stopped state. A filesystem partition must be stopped in order for you to write a transaction log to it. You can start the filesystem again using the -c option.

The -i option provides useful statistical information about a running filesystem. This option is so common that it assumes `/dev/etfs2`, thus saving you from having to enter the -d option before it. The information is displayed in following groups:

- Device
- Pools
- Counts
- Errors

Device

Name

Name of the device. The name usually encodes a part number or size.

Blocks

Number of blocks on the device.

Clusters/Block

Number of clusters to a block on the device.

Clustersize

Size of a cluster. Typically 1 KB or 2 KB.

Totalsize

Total size of the device, in bytes.

Pools

Clean

Number of erased blocks immediately ready for writing.

Spare

Number of spare blocks.

Filthy

Number of free blocks that are waiting to be erased and made clean.

Copyright © 2014, QNX Software Systems Limited

Inactive

Number of clusters not being used but trapped.

Xpool

Number of cache buffers.

Cache

Number of cluster cache buffers.

Counts

Erase

Number of erases on the part (while running).

Avg

Average erase count per block.

Read

Number of cluster reads from the device.

Cache

Number of cluster reads from cache.

Write

Number of cluster writes to the device.

Mine

Number of mining operations to recover dead space in a block. This is how inactive clusters create filthy blocks, which become clean after being erased.

Copy

Number of block-copy operations. Copies occur two ways: the first way is a read in a block that has a soft ECC error, which is an indication that the block is getting weak. The block is copied to a new fresh block and the block with the ECC error is erased. In the second way, a block with a low erase count is forced into service by copying its data to a new block and erasing and putting this block into service.

Defrag

Number of files defragmented.

BadBlks

Number of blocks marked as bad and taken out of service

Copyright © 2014, QNX Software Systems Limited

Errors

Ecc

Number of CRC data errors that are corrected by ECC.

Chksum

Number of CRC data errors.

Device

Number of hard device errors. This is bad and usually indicates a hardware problem.

The error statistics currently aren't collected, so these values are always 0.

Examples:

Print information about a filesystem. If you omit the -d option, `etfsctl` assumes `/dev/etfs2`.

```
etfsctl -i
etfsctl -d /dev/etfs2 -i
```

Format an empty filesystem:

```
etfsctl -d /dev/etfs2 -S -f -c
```

Write a filesystem built by *mketfs* (p. 1219):

```
etfsctl -d /dev/etfs2 -S -e -w fsys.etfs -c
```

Save the entire filesystem. Erase the part, and restore the filesystem:

```
etfsctl -d /dev/etfs2 -r debug.etfs
etfsctl -d /dev/etfs2 -S -e -w debug.etfs -c
```

Erase part of a raw ETFS partition, without erasing the boot monitor:

```
etfsctl -d /dev/etfs1 -o 3M -l 6M -e
```

Read part of a raw partition:

```
etfsctl -d /dev/etfs1 -o 3m -2k -R /temp/raw.stuff
```


Copyright © 2014, QNX Software Systems Limited

expand

Convert tabs to spaces (POSIX)

Syntax:

```
expand [-t tablist] [file...]
```

Runs on:

QNX Neutrino

Options:

-t tablist

Set up tab stops according to the *tablist* argument. This argument consists either of a single positive decimal integer, or of multiple positive decimal integers, in ascending order, separated by single commas.

If no number is given, tabs stops are set eight columns apart. If a single number is given, tab stops are set *tablist* columns apart. If multiple numbers are given, tab stops are set at those columns.

file

The pathname of a file whose tabs are to be converted.

Description:

The expand utility copies files or the standard input to the standard output with tab characters replaced by the number of spaces needed to pad to the next tab stop. Any backspace characters encountered in the input are copied to the output and cause the column position count for the tab-stop calculations to be decremented; the count is never decremented below zero.

The -t option lets you specify how many columns tab stops are set apart. You can also use it to specify a multiple tab-stop list that determines where tab stops are placed. If tab characters are present in the input past the last tab stop specified in a multiple tab-stop list, those tabs are each replaced by a single space in the output.

Note that tabbing to position *N* causes the next character written to appear in the *next* column position on that line (i.e. column *N*+1)

Copyright © 2014, QNX Software Systems Limited

Examples:

For the file `myfile`, expand each tab to the number of spaces required to reach the next tab stop:

```
expand myfile
```

Do the same as above, except set tab stops to every four columns rather than the default eight:

```
expand -t4 myfile
```

Place tab stops at the specified columns. Any tab encountered past the last tab stop is replaced by a single space:

```
expand -t8,12,20,24,32,36,44,48 myfile
```

Exit status:

0

Successful completion.

>0

An error occurred.

Caveats:

The `expand` utility doesn't check to verify that the tab stops specified in the -t option are in ascending order, as is required for proper operation.

Copyright © 2014, QNX Software Systems Limited

/etc/exports

Define remote mountpoints for NFS mount requests

Name:

`/etc/exports`

`/etc/exports.`*hostname*

Description:

The `exports` file defines remote mountpoints for the NFS mount protocol according to the NFS server specification; see *RFC 1094* (Network File System Protocol Specification) and *RFC 1813* (NFS Version 3 Protocol Specification).

There isn't a default version of this file; you can create your own if you need it.

Each line in the file specifies one remote mountpoint. The first field contains the mount-point directory path, followed optionally by a list of options and/or a list of specific hosts separated by whitespace. If no specific hosts are specified, the mount point is exported to all hosts.

Here are the export options:

-mask=*netmask* -match=*network*

Restrict access to hosts belonging to subnet defined by netmask and network. By default, there's no restriction. Access is determined by:

```
((client_ip & netmask) == network)
```

-norsvd

Don't check incoming requests, they're from a reserved port. By default, NFS requests from ports greater than `IPPROTO_RESERVED` are replied to with `EACCES`.

-ro

Export the filesystem as read-only. By default, the filesystem is exported as read/write.

-root=*uid*

Map root's uid (real user ID). By default, root is mapped to `-2`.

Copyright © 2014, QNX Software Systems Limited

Let's now look at a sample file:

```
/usr -root=1 rickers snowhite.cis.uoguelph.ca
/usr/local 131.104.48.16
/u -root=5 -mask=255.255.240.0 -match=131.104.0.0
/u2 -ro -mask=255.0.0.0 -match=10.0.0.0 node11 node23
```

The above example specifies the following:

This mountpoint:	Is exported:
`/usr`	To hosts `rickers` and `snowhite.cis.uoguelph.ca` only, with `root` mapped to `1` and with read/write access.
`/usr/local`	To host `131.104.48.16` only, with `root` mapped to `-2` and with read/write access.
`/u`	To all hosts within `131.104.0.0` to `131.104.15.255`, with `root` mapped to `5` and with read/write access.
`/u2`	To hosts `node11` and `node23` and to hosts belonging to IP network `10` only, with `root` mapped to `-2` and with read-only access.

Limitations:

1 subnet per mountpoint

10 hosts per mountpoint

Based on:

- *RFC 1094* (Network File System Protocol Specification)
- *RFC 1813* (NFS Version 3 Protocol Specification)

Copyright © 2014, QNX Software Systems Limited

expr

Evaluate arguments as an expression (POSIX)

Syntax:

`expr` *operand...*

Runs on:

QNX Neutrino

Options:

operand

Operand expression to evaluate.

Description:

The `expr` utility evaluates a single expression and writes the result to the standard output. The expression is formed from integer and string symbols in combination with the following operators:

```
(  )  |  &  =  >  >=  <  <=  !=  +  -  *  /  %  :
```

In the following table, expressions are listed in order of decreasing precedence, with equal-precedence operators grouped together. All of the operators are left-associative. Note that a string operand is an argument that can't be identified as an integer argument or as one of the above operators.

Expression	Description
integer	An argument consisting only of an (optional) unary minus followed by digits
string	A string argument
(*expr*)	Grouping symbols; you can place any expression inside the parentheses.
expr1 : *expr2*	Matching expression
expr1 * *expr2*	Integer multiplication
expr1 / *expr2*	Integer division, producing an integer result
expr1 % *expr2*	Remainder of integer division

Copyright © 2014, QNX Software Systems Limited

Expression	Description
expr1 + *expr2*	Integer addition
expr1 - *expr2*	Integer subtraction
expr1 = *expr2*	Equal*
expr1 > *expr2*	Greater than*
expr1 >= *expr2*	Greater than or equal*
expr1 < *expr2*	Less than*
expr1 <= *expr2*	Less than or equal*
expr1 != *expr2*	Not equal*
expr1 & *expr2*	Returns the evaluation of *expr1* if neither expression evaluates to null or zero; otherwise, returns zero
expr1 | *expr2*	Returns the evaluation of *expr1* if it's neither null nor zero; otherwise returns the evaluation of *expr2*

* Returns the result of a decimal integer comparison if both arguments are `integers`; otherwise, returns the result of a string comparison. The result of each comparison is 1 if the specified relation is TRUE or 0 if FALSE.

The "`:`" matching operator compares the string resulting from the evaluation of *expr1* with the regular expression pattern resulting from the evaluation of *expr2*. Usually, this operator returns the string representing the number of characters matched ("`0`" on failure). However, if the pattern contains at least one regular expression subexpression `\(...\)`, the string corresponding to `\1` is returned (see *grep* (p. 914)).

Examples:

Add 1 + 1 :

```
expr 1 + 1
```

Logical 1 `or` 0 :

```
expr 1 \| 0
```

Logical 1 `and` 0 :

```
expr 1 \& 0
```

Add 1 to $*a* :

```
expr $(expr $a + 1)
```


Copyright © 2014, QNX Software Systems Limited

Count the characters in $*var* :

```
expr "$var" : '.*'
```

Compare $*a* to a possible =

```
expr x$a = x=
```

Exit status:

0

The expression evaluates to neither null nor zero.

1

The expression evaluates to null or zero.

2

Invalid expressions were found.

>2

An error occurred.

Caveats:

The syntax for `expr` requires special attention. Many of the operators are also shell control operators or reserved words, so they have to be escaped on the command line. In addition, each part of the expression is composed of separate arguments, so you should use blanks liberally. For example:

Instead of entering:	You should enter:
`expr "1 + 2"`	`expr 1 + 2`
`expr 1 + (2 * 3)`	`expr 1 + \( 2 \* 3 \)`
`expr 1+2`	`expr 1 + 2`

In many cases, the arithmetic and string features provided as part of the shell command language are easier to use than their equivalents in `expr`. However, you may need `expr` to run older UNIX shell scripts.

When you want to specify the filesystem root, avoid the `/` character as a standalone argument — `expr` interprets it as the division operator. Use `//` instead.

Copyright © 2014, QNX Software Systems Limited

Chapter 7
F

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`flashcmp`	*`deflate`* (p. 183) and *`inflator`* (p. 984)

This chapter describes the utilities, etc. whose names start with “F”.

Copyright © 2014, QNX Software Systems Limited

false

Return false value (POSIX)

Syntax:

```
false
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

None.

Description:

The `false` utility does nothing but exit immediately with a nonzero exit code.

The shell has a builtin *`false`* (p. 1063) command; see *`ksh`* (p. 1029). To make sure you use the executable, specify the full path.

Exit status:

> 0

Copyright © 2014, QNX Software Systems Limited

fcat

Display compressed (frozen) files (UNIX)

Syntax:

```
fcat [filename...]
```

Runs on:

QNX Neutrino

Options:

See *freeze* (p. 784) for a full listing.

Description:

The `fcat` command is equivalent to `freeze -cd`. See the *freeze* (p. 784) utility for details.

The `freeze`, *melt* (p. 1197), and `fcat` compression utilities will eventually become deprecated in favor of the GNU `zip` suite, which consists of *gzip* (p. 921), *gunzip* (p. 920), and *zcat* (p. 2080). The `freeze` suite of utilities will continue to be provided for quite some time before being eliminated completely.

Contributing author:

Leonid A. Broukhis

Copyright © 2014, QNX Software Systems Limited

fdformat

Format floppy diskettes (QNX)

Syntax:

```
fdformat [-aiIpqv] [-h heads] [-t tracks] [-n sectors]
         [-s size] [-k skew_factor] [-z interlv] drive
```

Runs on:

QNX Neutrino

Options:

-a

Abort on the first error encountered.

-h *heads*

The number of heads.

-I

Ignore errors. Don't try to verify diskette.

-i

Write layout information to the diskette, but don't format it.

-k *skew_factor*

The number of sectors by which `fdformat` should offset the starting sector of a track from the last sector of the previous track.

-n *sectors*

The number of sectors per track.

-p

Prompt before starting.

-q

Be quiet; don't report progress as each track is formatted.

-s *size*

Copyright © 2014, QNX Software Systems Limited

The disk media: 360K, 720K, 1.2M, 1.4M, or 2.8M (overrides -h, -t, and -n).

-t *tracks*

The number of tracks.

-v

Be verbose; display extra information as the format progresses.

-z *interlv*

The interleave amount.

drive

The name of the physical device to be formatted (i.e. `/dev/fd0`).

Description:

The `fdformat` utility formats the specified disk. Formatting refers to a process of placing addressing marks and other control information on the disk to allow the hardware to read it. Formatting doesn't imply any sort of filesystem structure.

By default, `fdformat` uses the current mount characteristics of the drive as the drive parameters. These parameters may be overridden by specific command-line options, such as -t *tracks*.

The -s *size* option lets you specify the size of standard floppy disks, as in the following:

Size	Heads	Tracks	Sectors	Drive type
360K	2	40	9	5¼ floppy
720K	2	80	9	3½ floppy
1.2M	2	80	15	5¼ floppy
1.44M	2	80	18	3½ floppy
2.88M	2	80	36	3½ floppy

The first sector of QNX Neutrino floppy disks contains some information about the layout of the diskette (heads, tracks, sectors). The -i option simply writes this layout information to a diskette without formatting it.

The `-z` *interlv* option lets you specify the amount of interleave (spacing between sectors). Specifying -z1 (the default) would place the sectors contiguously (e.g. 1,2,3,4,5, etc.)

Specifying -z2 would place the sectors at every second location (e.g. 1, -, 2, -, 3, -, 4, -, 5, etc.). If there were nine sectors per track, -z2 would yield:

Copyright © 2014, QNX Software Systems Limited

```
1, 6, 2, 7, 3, 8, 4, 9, 5
```

such that reading every other sector in the (circular) track would produce a contiguous reading of the sectors. Note that the hardware makes all of this transparent; specifying -z may optimize access for the hardware.

The -k *skew_factor* option affects performance when reading and writing consecutive tracks on the disk in an ascending order. Disk drives have an inherent latency when moving the head from one track to the next. By setting the first sector of the next track some distance ahead of the last sector of the current track, it is possible to tune the amount of time it takes before the head encounters the first sector of the next track after the seek. When there is no skew, the rotation of the disk places the head past the first sector of the next track after the time taken to seek to the next track, meaning that to get to the first sector the disk must complete nearly a full rotation.

If the latency is, for example, five sectors, you would specify the following:

```
-k 5
```

Placing the first sector of the next track five sectors away from the current sector minimizes the latency effects.

Examples:

Format the diskette mounted as `/dev/fd0` using the current mount parameters:

```
fdformat /dev/fd0
```

Format a 1.4M diskette mounted as `/dev/fd0`:

```
fdformat -s 1.4M /dev/fd0
```

Format a 1.4M diskette mounted as `/dev/fd0` using an interleave of 3 because the diskette will be used as a boot diskette:

```
fdformat -s 1.4M -z 3 /dev/fd0
```

Files:

If the -p is specified, `fdformat` pauses until gets a newline character from the standard input before proceeding to format the specified disk. The standard input is otherwise unused.

If not in quiet mode (i.e. -q isn't specified), informational and progress messages are written to the standard output as the formatting proceeds. The extent of this information is influenced by the -v option. If -q is specified, this information isn't written. Note that -q doesn't override the printing of the prompt for the -p (pause) option.

Any errors which occur cause a diagnostic message to be written to the standard error. In addition, if the -v option is specified, information relating to some of the specific formatting steps may be written to the standard error.

Copyright © 2014, QNX Software Systems Limited

The `fdformat` utility acts upon the block special file named on the command line. If successful, `fdformat` destroys the contents of the medium represented by this file.

Exit status:

0

The disk was formatted successfully.

>0

An error occurred and a diagnostic message was written to the standard error.

Caveats:

The `fdformat` utility destroys any existing data on a diskette. Don't count on this behavior for data security — differences in physical drive characteristics and presence of magnetic fields in fringe areas may result in the data being recoverable by use of special instruments. If you want to destroy data, the only really safe way is to completely destroy the media.

Note that the latency saved by tuning the disk skew factor is only realized when reading sequentially across a track boundary. On random requests, there is no single mechanism to minimize access time.

Copyright © 2014, QNX Software Systems Limited

fdisk

Create and manage partitions on a hard disk

In order to run this utility, you must be logged in as `root` or have read/write permissions for the block-special file concerned.

Syntax:

```
fdisk [-fz] [-B loader] drive [cmd [args]]
```

Runs on:

QNX Neutrino

Options:

-B *loader*

Use the 512-byte file named by *loader* as the primary bootstrap loader for the device when instructed to write a boot loader to the disk. The default is to install a loader that's built into the `fdisk` utility.

-f

Force the boot loader to be written on command, even if it isn't possible to save an existing old loader to a mounted filesystem. In non-interactive mode, execute the given `add` command, even if this would involve overwriting an existing partition slot.

-z

Zero the partition table (interactive mode only).

drive

The disk drive to partition. This must name a block-special file (e.g. `/dev/fd1`, `/dev/hd0`).

***cmd* [*args*]**

An installation command, as described *below* (p. 736).

Description:

The `fdisk` utility lets you create and manage partitions on a hard disk (typically a rotating medium, but `fdisk` works on other devices, such as compact flash and USB

Copyright © 2014, QNX Software Systems Limited

flash, if they support PC-style Master Boot Records (MBRs) and partitions). The partition information, which is kept in the disk's first physical block, matches that used by DOS.

On some platforms, `fdisk` supports a full-screen interface; see "*Interactive mode* (p. 739)," below.

- The installer for Microsoft Windows overwrites any existing Master Boot Record with its own. If you want your disk to contain bootable DOS and bootable QNX (or other non-DOS) partitions, you should install Windows *first*, and then create the other partitions. If you create a QNX partition first and then install Windows, you can restore the boot loader by running QNX Neutrino from the installation disk and explicitly using *`dloader`* (p. 633).
- On some x86 machines, you can boot only from OS images that are loaded from within the first 1024 cylinders of the disk. This means that while you may be able to initially install and boot from a partition which extends past the 1024th cylinder, it will someday fail when you go to update the boot image because the location of some of its blocks may change. When this happens you will have a system that's no longer bootable.

  Avoid this problem by creating a separate partition to boot from that lies entirely within the first 1024 cylinders of the hard drive, and use a second partition to access the additional space on the drive. (The boot partition may be quite small — just a few megabytes will suffice.)

Before creating a partition for the first time, you must start the hard disk driver:

```
devb-eide &
```

You should then execute the `fdisk` command to partition your disk. For example:

```
fdisk /dev/hd0 add -t 179 -p 50
```

New or changed partitions aren't immediately recognized and/or mounted. You must either slay and restart the filesystem/driver (*`devb-*`* (p. 169)), use *`mount -e /dev/hd0`* (p. 1312) to recognize the new partitions and update the contents of `/dev`, or reboot.

Partition types

The `fdisk` utility recognizes the following partition types. If you add a partition, use the command shown to initialize it.

Copyright © 2014, QNX Software Systems Limited

Type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
5	DOS extended	N/A	N/A	N/A
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
8 or 9	QNX 2	N/A	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
15	Windows 95 extended	N/A	N/A	N/A
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
99	UNIX	N/A	N/A	N/A
130	Linux swap	N/A	N/A	N/A
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
133	Linux extended	N/A	N/A	N/A
165	BSD	N/A	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Commands

The `fdisk` utility supports the following commands directly from the command line:

add [*args*]

Add a new partition entry of the size and type specified. If `fdisk` can't locate sufficient unallocated disk space to satisfy your request, it allocates

Copyright © 2014, QNX Software Systems Limited

the largest available portion of the disk (if any). Here are the arguments for `add`:

-b

Make the added partition bootable. If another partition was already flagged as the primary boot partition, the flag is turned off for it.

-c *start,end*

The start and end for the partition to use.

-e *extended_slot*

The index into the extended partition (1–*N*).

-n *count*

The number of cylinders to allocate (extended partitions only).

-p *percent*

The percentage of the largest contiguous space the added partition should use. The default is 100%.

If you specify the -c option, the -p option is ignored.

-s *slot*

The slot (1-4) in the partition table to use. The default is the last free slot.

-t *type*

The type of partition to add (0 - 255). The default is 77.

Note that extended partition indexes may change due to additions.

`boot` [*args*]

Turn on the boot flag for the indicated partition. If another partition was already flagged as the primary boot partition, the flag is turned off for it. Here are the arguments for `boot`:

-e *extended_slot*

The index into the extended partition (1–*N*).

-s *slot*

Boot the partition in the selected slot.

Copyright © 2014, QNX Software Systems Limited

-t *type*

Boot the partition of the selected type.

`delete [`*`args`*`]`

Delete the specified partitions. Here are the arguments for `delete`:

-a

Delete all partitions.

-e *extended_slot*

The index into the extended partition (1–*N*).

-s *slot*

Delete the partition in the selected slot.

-t *type*

Delete this *type* of partition.

Note that extended partition indexes may change due to deletions.

`info`

Show the mount information for the raw drive.

The `fdisk` utility makes a `devctl(DCMD_CAM_DEVINFO)` call to obtain the cylinder, head, sectors per track, and total sectors counts. Multiplying the first three values together is the classic method of calculating the total number of sectors.

However, some hard drives employ zoned bit recording, so it's impossible to precisely map the number of sectors per track and other fields. As a result, the total number of sectors returned from *devctl()* and the total number of sectors that were calculated might not match. In this case, `fdisk` displays a warning.

`loader`

Write the QNX loader to the disk.

`query [`*`args`*`]`

Print the number of cylinders to standard output. Here are the arguments for `query`:

-e *extended_slot*

Copyright © 2014, QNX Software Systems Limited

The index into the extended partition (1–*N*).

-f

Display the total number of free cylinders.

-s *slot*

Query the partition in the selected slot.

-T

Display the total number of cylinders.

-t *type*

Query the partition of the selected type.

show

Display the partition table.

Interactive mode

On some platforms, `fdisk` is a fullscreen, interactive program that's fairly self-explanatory. When you invoke `fdisk`, you'll see a screen similar to this one (assuming your disk is already partitioned):

```
FDISK
Ignore Next Prev 1 2 3 4 Change Delete Boot Unboot Restore Loader Save Quit

        _____OS_____      Start      End      ______Number_____     Size    Boot
        name    type     Cylinder  Cylinder  Cylinders  Blocks

</comment> 1.  QNX6   (177)          0      7648      7649   122881122  60000 MB
    2.  QNX6   (178)       7649      9963      2315    37190475  18159 MB  *
    3.  ______ (___)    ________  ________  ________  _________  _____
    4.  ______ (___)    ________  ________  ________  _________  _____


 Choose a partition by typing the partition number OR moving the pointer
 with the UP/DOWN arrows.
 Then, choose one of the actions on the top line of the screen.


Drive : /dev/hd0                    Config:   255 Heads
Size  : 78159 Mbytes                            63 Sectors/track
Loader: Unknown                               9964 Cylinders
                                               512 Block Size

                                    Last cylinder is 9963
```

You'll see the available commands displayed at the top of the screen. To select a command, either type its first letter or move the cursor to the command (with the arrow keys) and press **Enter**.

The commands are:

Command:	Action:
`Next`	Move the pointer to the next entry.
`Prev`	Move the pointer to the previous entry.

Copyright © 2014, QNX Software Systems Limited

Command:	Action:
`1`, `2`, `3`, or `4`	Move the pointer to the indicated entry.
`Change`	Change the selected partition (see below).
`Delete`	Delete the selected partition.
`Boot`	Turn on the boot flag for the selected partition. If another partition was already flagged as the primary boot partition, the flag is turned off for it.
`Unboot`	Turn off the boot flag for the selected partition.
`Restore`	Restore the previous non-QNX bootstrap loader.
`Loader`	Change the bootstrap loader to the QNX loader.
`Save`	Save all changes and quit. This writes to the device and is irrevocable.
`Quit`	Quit without saving changes.

If you're changing a partition entry, note the following:

- Save the details about the partition (e.g. by writing them on a piece of paper), because `fdisk` blanks the fields as you edit them.
- You have to enter the partition's type number and the start and end cylinders; `fdisk` calculates the other information for you. Press **Enter** after typing each value.
- If the partition was bootable before you changed it, use the `Boot` command to make it bootable again.

Examples:

Create a QNX 4 partition that occupies half the disk, or the largest available space if there isn't a space big enough for a new partition that occupies half the disk:

```
fdisk /dev/hd0 add -t 77 -p 50
```

Do the same, but make the partition bootable:

```
fdisk /dev/hd0 add -b -t 77 -p 50
```


Copyright © 2014, QNX Software Systems Limited

Continuing from either of the above examples, reread the partition table, set up a QNX 4 filesystem on the new partition, and then mount it:

```
mount -e /dev/hd0
dinit -h /dev/hd0t77
mount -t qnx4 /dev/hd0t77 /mnt/q4fs
```

Create a bootable partition for a Power-Safe filesystem, reread the partition table, format the new partition, and then mount it:

```
fdisk /dev/hd0 add -b -t 179 -p 50
mount -e /dev/hd0
mkqnx6fs /dev/hd0t179
mount -t qnx6 /dev/hd0t179 /mnt/psfs
```

Exit status:

0

Success.

>0

An error occurred; `fdisk` writes error messages to standard error.

Caveats:

After changing *any* partition information, you must either slay and restart the filesystem/driver (*devb-** (p. 169)) or use *mount -e* (p. 1312) to make the filesystem reread the partition table.

Copyright © 2014, QNX Software Systems Limited

fesh

Fat embedded shell (QNX Neutrino)

Syntax:

```
fesh [-c command] [-irv] [script_file]
```

Runs on:

QNX Neutrino

Options:

-c *command*

Run this command.

-i

Enter interactive mode after running a script file. If you don't specify this option, `fesh` terminates after running a script.

-r

Run in restricted mode. In this mode, you can't perform certain operations. If you try to, you'll see the error message, "Operation not permitted."

Restricted operations include running executables that start with a slash, exporting variables, and reattaching standard input, output and error to another device. For more information, see "*Command-line format* (p. 705)" and "*Builtin commands* (p. 707)" in the description of *esh* (p. 705).

-v

Be verbose: echo each command before executing it.

script_file

A file containing shell commands to execute.

Description:

The `fesh` utility is a "fat" version of the small embedded shell, *esh* (p. 705). The `fesh` shell supports all of the *esh builtin commands* (p. 707), as well as the following extra builtin commands:

- *ecp* (p. 743)

Copyright © 2014, QNX Software Systems Limited

- *edf* (p. 743)
- *eecho* (p. 743)
- *els* (p. 743)
- *emkdir* (p. 743)
- *epwd* (p. 743)
- *erm* (p. 743)
- *ermdir* (p. 744)

If fesh doesn't recognize the arguments to a builtin command, it searches for an executable file.

ecp command

```
ecp [-v] source destination
```

Copy *source* to *destination*, like the *cp* (p. 141) command. The -v option gives verbose output.

edf command

```
edf [path]
```

Report the free disk space for the filesystem associated with *path*. If you don't specify a path, fesh uses the current directory.

eecho command

```
eecho arguments
```

Write arguments to standard output, like *echo* (p. 668). No options are defined.

els command

```
els [-l] [path]
```

List the contents of a directory, like *ls* (p. 1139). The els command supports only the -l option, and displays the permissions in octal.

emkdir command

```
emkdir dir
```

Create a directory, like *mkdir* (p. 1202). No options are defined.

epwd command

```
epwd
```

Print the name of the current working directory, like the *pwd* (p. 1602) executable.

erm command

```
erm file...
```

Remove the specified file or files. No options are supported.

Copyright © 2014, QNX Software Systems Limited

The `erm` builtin command behaves like *rm* (p. 1673) -f.

`ermdir` command

```
ermdir dir...
```

Similar to *rmdir* (p. 1675) but doesn't support any options.

Copyright © 2014, QNX Software Systems Limited

fgrep

Fixed string `grep` (UNIX)

Syntax:

```
fgrep [-cilnqsvx]
      [-e expression | -f expression_file]...
      [file...]
fgrep [-cilnqsvx] expression [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

See *grep* (p. 914) for a complete listing.

Description:

The `fgrep` utility does a fixed-string `grep` (equivalent to `grep -F`). See the documentation for `grep` for details.

Exit status:

0

Lines were found matching the expression provided.

>0

An error occurred or no matching lines were found.

Copyright © 2014, QNX Software Systems Limited

file

Determine file type (UNIX)

Syntax:

```
file [-bcLnvz] [-f namefile] [-m magicfile] file ...
```

Runs on:

QNX Neutrino

Options:

-b

Don't echo the name of the file before its type.

-c

Cause a checking printout of the parsed form of the `magic` file. This is usually used in conjunction with -m to debug a new magic file before installing it.

-f *namefile*

Read the names of the files to be examined from *namefile* (one per line) before the argument list. Either *namefile* or at least one filename argument must be present; to test the standard input, use - as a filename argument.

-L

Cause symlinks to be followed, as the like-named option in *ls* (p. 1139).

-m *file*

Specify an alternate file of magic numbers. The default is */usr/share/misc/magic* (p. 1163).

-n

Echo the name of the file before its type (this is done by default).

-v

Print the version of the program and exit.

-z

Try to look inside compressed files.

Copyright © 2014, QNX Software Systems Limited

Description:

The `file` utility tests each *file* argument in an attempt to classify it. There are three sets of tests, performed in this order:

1. Filesystem tests
2. Magic number tests
3. Language tests

The *first* test that succeeds causes the file type to be printed. The type printed usually contains one of these words:

`text`

The file contains only ASCII characters and is probably safe to read on an ASCII terminal.

`executable`

The file contains the result of compiling a program in a form understandable to some UNIX kernel or another.

`data`

Meaning anything else (data is usually "binary" or nonprintable). Exceptions are well-known file formats (`core` files, `tar` archives) that are known to contain binary data.

When modifying the file */usr/share/misc/magic* (p. 1163) or the program itself, *preserve these keywords*. People depend on knowing that all the readable files in a directory have the word `text` printed.

The filesystem tests are based on examining the return from a *stat()* system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you're running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they're defined in the system header file `/usr/include/sys/stat.h`.

The magic number tests are used to check for files with data in particular fixed formats. These files have a "magic number" stored in a particular place near the beginning of the file that tells the UNIX operating system that the file is a binary executable, and which of several types thereof. The concept of "magic number" has been applied by extension to data files. Any file with some invariant identifier at a small fixed offset into the file can usually be described in this way. The information in these files is read from the magic file `/usr/share/misc/magic`.

If an argument appears to be an ASCII file, `file` attempts to guess its language. The language tests look for particular strings that can appear anywhere in the first few blocks of a file. For example, the keyword `.br` indicates that the file is most likely a

Copyright © 2014, QNX Software Systems Limited

`troff` input file, just as the keyword `struct` indicates a C program. These tests are less reliable than the previous two groups, so they are performed last. The language test routines also test for some miscellany (such as `tar` archives) and determine whether an unknown file should be labeled as `ascii text` or `data`.

Files:

/usr/share/misc/magic **(p. 1163)**

Default list of magic numbers.

Contributing author:

Written by Ian F. Darwin, UUCP address `{utzoo|ihnp4}!darwin!ian`, Internet address `ian@sq.com`, postal address: P.O. Box 603, Station F, Toronto, Ontario, CANADA M4Y 2L8.

Altered by Rob McMahon, `cudcv@warwick.ac.uk`, 1989, to extend the & operator from simple `x&y != 0` to `x&y op z`.

Altered by Guy Harris, `guy@auspex.com`, 1993, to:

- Put the "old-style" & operator back the way it was, because 1) Rob McMahon's change broke the previous style of usage, 2) the SunOS "new-style" & operator, which this version of `file` supports, also handles `x&y op z`, and 3) Rob's change wasn't documented in any case;
- Put in multiple levels of >
- Put in `beshort`, `leshort`, etc. keywords to look at numbers in the file in a specific byte order, rather than in the native byte order of the process running `file`.

Changes by Ian Darwin and various authors including Christos Zoulas (`christos@ee.cornell.edu`), 1990-1992.

License:

Copyright © Ian F. Darwin, Toronto, Canada, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993.

This software is not subject to and may not be made subject to any license of the American Telephone and Telegraph Company, Sun Microsystems Inc., Digital Equipment Inc., Lotus Development Inc., the Regents of the University of California, The X Consortium or MIT, or The Free Software Foundation.

This software is not subject to any export provision of the United States Department of Commerce, and may be exported to any country or planet.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

Copyright © 2014, QNX Software Systems Limited

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

A few support files (*getopt()*, *strtok()*) distributed with this package are by Henry Spencer and are subject to the same terms as above. A few simple support files (*strtol()*, *strchr()*) distributed with this package are in the public domain; they are so marked.

The files `tar.h` and `is_tar.c` were written by John Gilmore from his public-domain `tar` program, and are not covered by the above restrictions.

Caveats:

The `file` utility uses several algorithms that favor speed over accuracy, thus it can be misled about the contents of ASCII files. The support for ASCII files (primarily for programming languages) is simplistic, inefficient and requires recompilation to update.

Copyright © 2014, QNX Software Systems Limited

find

Find files (POSIX)

Syntax:

```
find path... [operand_expression]
find [limited_operand_expression]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

path...

Pathnames under which `find` should search for files. The utility traverses the pathname tree from these files down, looking for files that match the search criteria specified by the *operand_expression*. If no paths are specified, and the expression meets the criteria of a *limited_operand_expression* (below), then a path of `.` is assumed.

operand_expression

An expression composed of any set of the *primary expressions* (p. 752) and operators described below.

Here are some links to help you find the expressions:

-abort (p. ?) ¦ *-amin* (p. ?) ¦ *-anewer* (p. ?) ¦ *-atime* (p. ?) ¦ *-chgrp* (p. ?) ¦ *-chmod* (p. ?) ¦ *-chown* (p. ?) ¦ *-cmin* (p. ?) ¦ *-cnewer* (p. ?) ¦ *-ctime* (p. ?) ¦ *-daystart* (p. ?) ¦ *-depth* (p. ?) ¦ *-echo* (p. ?) ¦ *-empty* (p. ?) ¦ *-errmsg* (p. ?) ¦ *-error* (p. ?) ¦ *-exec* (p. ?) ¦ *-exists* (p. ?) ¦ *-false* (p. ?) ¦ *-fanewer* (p. ?) ¦ *-fcnewer* (p. ?) ¦ *-fls* (p. ?) ¦ *-fmnewer* (p. ?) ¦ *-fnewer* (p. ?) ¦ *-Fnewer* (p. ?) ¦ *-follow* (p. ?) ¦ *-fprint* (p. ?) ¦ *-fprint0* (p. ?) ¦ *-fprintf* (p. ?) ¦ *-gid* (p. ?) ¦ *-group* (p. ?) ¦ *-ilname* (p. ?) ¦ *-iname* (p. ?) ¦ *-inode* (p. ?) ¦ *-inum* (p. ?) ¦ *-ipath* (p. ?) ¦ *-iregex* (p. ?) ¦ *-level* (p. ?) ¦ *-links* (p. ?) ¦ *-lname* (p. ?) ¦ *-logical* (p. ?) ¦ *-ls* (p. ?) ¦ *-maxdepth* (p. ?) ¦ *-mindepth* (p. ?) ¦ *-mmin* (p. ?) ¦ *-mnewer* (p. ?) ¦ *-mount* (p. ?) ¦ *-mtime* (p. ?) ¦ *-name* (p. ?) ¦ *-newer* (p. ?) ¦ *-nogroup* (p. ?) ¦ *-NOP* (p. ?) ¦ *-nouser* (p. ?) ¦ *-ok* (p. ?) ¦ *-path* (p. ?) ¦ *-perm (symbolic)* (p. ?) ¦ *-perm (octal)* (p. ?) ¦ *-pname* (p. ?) ¦ *-print* (p. ?) ¦ *-print0* (p. ?) ¦ *-printf* (p. ?) ¦ *-prune* (p. ?) ¦ *-regex* (p. ?) ¦

Copyright © 2014, QNX Software Systems Limited

-remove! (p. ?) ¦ *-rename* (p. ?) ¦ *-size* (p. ?) ¦ *-spawn* (p. ?) ¦ *-true* (p. ?) ¦ *-type* (p. ?) ¦ *-uid* (p. ?) ¦ *-used* (p. ?) ¦ *-user* (p. ?) ¦ *-xdev* (p. ?)

limited_operand_expression

An expression composed of any of the primary expressions and operators described below, *except* for *-exec* (p. ?), *-ok* (p. ?), and *-spawn* (p. ?).

Description:

The `find` utility recursively descends the directory hierarchy for each file specified by *path* and seeks files that match *operand_expression*.

If you don't specify an *operand_expression* or *limited_operand_expression* on the command line, `find` uses *-print* (p. ?) (i.e. the utility matches every file and directory, printing each pathname on its own line to standard output).

The operand expression follows one or more pathnames. The `find` utility treats as a pathname all the arguments up to the first one starting with any of these characters:

```
-  !  (
```

Everything after that is part of the operand expression.

You'll probably have to quote some operands and patterns, depending on the shell that you're using.

Operand expressions are made up of primaries and operators. The following operators are supported:

Operator	Action
!	(NOT)
-a	(AND)
-o	(OR)

AND operations have higher precedence than OR operators. Negation (!) has a higher precedence than AND operators. Parentheses are supported to override the normal precedence rules.

The rules that apply to primaries and operators are:

Expression	Evaluates to
-*primary*	True if *primary* is true.
(*expression*)	True if *expression* is true.

Copyright © 2014, QNX Software Systems Limited

Expression	Evaluates to
! *expression*	(NOT) Negation of a primary or expression enclosed in parentheses.
expression [-a] *expression*	(AND) True if both expressions are true. If the first expression is false, the second expression isn't evaluated. The -a is optional. AND is implied by the juxtaposition of two expressions (see below).
expression -o *expression*	(OR) True if either expression is true. If the first expression is true, the second expression isn't evaluated.

As mentioned above, the -a operand is optional. If you want to match files of two different patterns, you might be inclined to use this command:

```
find . -name "*~" -o -name "*.o" -print
```

but this doesn't work the way you might expect it to because of the implicit -a before the -print expression. The rules of precedence make the above command equivalent to this:

```
find . \( -name "*~" \) -o \( -name "*.o" -a -print \)
```

You should specify the command like this:

```
find . \( -name "*~" -o -name "*.o" \) -print
```

Primary expressions

Note that if you don't supply an expression, find behaves as if you specified *-print* (p. ?).

If you specify an expression, but it doesn't contain a *-chmod* (p. ?), *-chown* (p. ?), *-exec* (p. ?), *-fls* (p. ?), *-fprint* (p. ?), *-fprint0* (p. ?), *-fprintf* (p. ?), *-ls* (p. ?), *-ok* (p. ?), *-print* (p. ?), *-print0* (p. ?), *-printf* (p. ?), *-rename* (p. ?), *-remove!* (p. ?), or *-spawn* (p. ?) primary, the find utility operates as if you specified the following expression:

```
( given_expression ) -print
```

Whenever a primary expression uses a number (*n*), you can optionally precede it by a plus (+) or minus (-) sign, which changes the meaning as follows:

Expression	Means
+*n*	More than *n*

Copyright © 2014, QNX Software Systems Limited

Expression	Means
-*n*	Less than *n*
n	Exactly *n*

We've classified the primary expressions as follows, in case you're interested in using `find` in a portable manner:

POSIX

Supported by any POSIX `find` implementation.

GNU

Supported by the QNX Neutrino and GNU `find` implementations.

QNX Neutrino

Supported only by the QNX Neutrino implementation.

The primary expressions are:

-abort

(QNX Neutrino) Immediately terminate the `find` with a nonzero exit status.

-amin *n*

(GNU) True if the file was last accessed *n* minutes ago.

-anewer *file*

(GNU) True if the file being evaluated was accessed more recently than *file*.

-atime *n*

(POSIX) True if the file access time subtracted from the time that the `find` utility started running is between *n*-1 and *n* multiples of 24 hours. For example, `find . -atime 1` finds all files under the current directory for which the file was accessed within the last 24 hours.

-chgrp *gname*

(QNX Neutrino) Change the file group ownership of the file currently being evaluated to *gname*. If *gname* is numeric and the name doesn't exist in the group database, *gname* is taken as a group ID.

Specifying -chgrp inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-chmod *mode*

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino) Change the permissions of the file currently being evaluated according to the specified *mode*.

The *mode* argument represents file mode bits. It's identical to the *symbolic_mode* operand described in *chmod* (p. 124) and is interpreted as follows.

A file *mode* has the form:

who{*op*}*perm*[,*mode*...]

where

- *who* may be

who	Permissions for:
u	The user who owns the file
g	The owner's group
o	Others
a	All users (equivalent to ugo)

  You can combine these. For example, if you specify ug, the subsequent *op* and *perm* arguments are applied to the user and group permissions on the file.

- *op* may be:

op	Action
+	Sets the appropriate mode bits
-	Clears the appropriate mode bits
=	Sets the appropriate mode bits, regardless of the process's file mode creation mask

- and *perm* may be any combination of

perm	Meaning
r	Read permission
w	Write permission (create/unlink for directories)
x	Execute permissions (search for directories)

Copyright © 2014, QNX Software Systems Limited

perm	Meaning
s	Setuid for owner or setgid for group, sticky for directories

For example, to remove write permission for group and other from the file being evaluated, use `-chmod go-w`.

Specifying -chmod inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-chown *uname*[:*gname*]

(QNX Neutrino) Change the file ownership of the file currently being evaluated to the one specified, taking a ownership specification similar to that accepted by the *chown* (p. 129) utility. A user name *uname* must be specified, optionally followed by a colon (`:`) and group name *gname*. The *uname* and *gname* parameters may be either an ASCII name or the actual numeric user or group ID.

Specifying -chown inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-cmin *n*

(GNU) True if the file status was last changed *n* minutes ago.

-cnewer *file*

(GNU) True if the file being evaluated had its status changed more recently than that of *file*.

-ctime *n*

(POSIX) True if the file change of status time subtracted from the time that the `find` utility started running is between *n*-1 and *n* multiples of 24 hours.

-daystart

(GNU) Always true. When used, this primitive causes `find` to globally alter the behavior of the *-atime* (p. ?), *-ctime* (p. ?), and *-mtime* (p. ?) primitives: instead of comparing file times to *n* 24-hour periods before the current time, it compares file times to *n* 24-hour periods before the beginning of the current calendar day.

-depth

(POSIX) Always true; causes descent of the directory hierarchy to be done so that all entries in the directory are acted on before the directory itself. If

Copyright © 2014, QNX Software Systems Limited

no -depth primary is specified, all entries in the directory are acted on after the directory itself. If any -depth primary is specified, it applies to the entire expression even if the -depth primary isn't normally evaluated.

-echo [*text*] ;

(QNX Neutrino) Write the supplied text to standard output. If the text contains any braces (`{}`), they're interpreted as in the *-exec* (p. ?) primitive to represent the pathname being evaluated.

Specifying -echo inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-empty

(GNU) True if the file is a regular file of size 0, or a directory that contains no files.

-errmsg [*text*] ;

(QNX Neutrino) Similar to *-echo* (p. ?), except the output is written to standard error.

Specifying -errmsg inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-error

(QNX Neutrino) Cause the exit status to be nonzero when the find is completed. This primary always evaluates to false and is typically used with *-exec* (p. ?). For example, if you enter:

```
find /bin -type f \( -exec cmp {} /hd{} \; \
     -o -error \)
```

`find` exits with a nonzero status if any of the files in `/bin` don't compare successfully against the same files under `/hd/bin`.

-exec *utility_name* [*argument...*] ;

(POSIX) True if the executed utility *utility_name* returns a zero value as exit status. The end of the primary expression is punctuated by a semicolon (`;`).

If a *utility_name* or *argument* contains `{...}`, the `{...}` is replaced by the current pathname or a portion thereof as follows:

A `{}` in a *utility_name* or *argument* is replaced by the pathname being evaluated. Such arguments are used by *-echo* (p. ?), *-errmsg* (p. ?), *-exists* (p. ?), *-fanewer* (p. ?), *-fcnewer* (p. ?), *-fmnewer* (p. ?), *-fnewer* (p. ?), *-ok* (p. ?), *-rename* (p. ?) and *-spawn* (p. ?) in addition to -exec.

Copyright © 2014, QNX Software Systems Limited

The current directory for the execution of *utility_name* is the same as the current directory when the `find` utility was started.

Specifying -exec inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

There is a QNX Neutrino-only extension to the `{}` syntax for stripping leading and trailing characters. You may also opt to insert the filename stripped of a number of characters at the end (*strip*) or the filename less a number of characters at the beginning (*skip*). The syntax for this is

```
{[strip][,skip]}
```

So, to move all files ending in `.c` to the same names ending in `.C`, one might use:

```
find . -type f -name '*.c' \
       -exec 'mv {} {1}C' \;
```

-exists *filestring*

(QNX Neutrino) True if the file represented by *filestring* exists. The *filestring* may be a simple filename or it may contain braces (`{}`) to represent the name of the file currently being evaluated, in the same manner as used with the *-exec* (p. ?) primitive.

-false

(GNU) Always false.

-fanewer *filestring*

(QNX Neutrino) True if the file was accessed more recently than the file represented by *filestring*.

-fcnewer *filestring*

(QNX Neutrino) True if the file had its status changed more recently than the file represented by *filestring*.

-fls *file*

(GNU) Always true. Similar to *-ls* (p. ?), but output is written to *file* instead of the standard output.

Specifying -fls inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-fmnewer *filestring*

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino) True if the file was modified more recently than the file represented by *filestring*.

-fnewer *filestring*

(QNX Neutrino) Synonym for *-fmnewer* (p. ?) (above).

-Fnewer *file*

(QNX Neutrino) True if the file being evaluated was created more recently than *file*.

-follow

(GNU) Always true. When -follow is specified, `find` treats symbolic links as being the type of the file they point to. If the link points to a directory, `find` recurses into that directory. By default, symbolic links are treated as special files of type symbolic link. What they point to is irrelevant to `find`'s default behavior.

-fprint *file*

(GNU) Similar to *-print* (p. ?), but output is written to *file* instead of to the standard output.

Specifying -fprint inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-fprint0 *file*

(GNU) Similar to *-print* (p. ?), but each file name is followed by a NUL instead of a newline character, and the output is written to *file* instead of to the standard output.

Specifying -fprint0 inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-fprintf *file format*

(GNU) Always true. Write data pertaining to the file currently being evaluated to *file*, according to the *format* specified. See "*Formatted Printing* (p. 765)" for details on the format string.

Specifying -fprintf inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-gid *n|groupname*

(GNU) Synonym for *-group* (p. ?).

Copyright © 2014, QNX Software Systems Limited

-group *gname*

(POSIX) True if the file belongs to the group *gname*. If *gname* is numeric and the name doesn't exist in the group database, *gname* is taken as a group ID. Note that *gname* is evaluated only once.

-ilname *fpattern*

(GNU) Similar to *-lname* (p. ?), but case-insensitive in the pattern match.

-iname *fpattern*

(GNU) Similar to *-name* (p. ?), but case-insensitive in the pattern match.

-inode *file|n*

(QNX Neutrino) True if the file uses the same inode (has the same serial number) as the named *file*. If the file doesn't exist and is numeric, the number is used as the serial number to match. This primary is used to find links to a file.

-inum *n|file*

(GNU) Synonym for *-inode* (p. ?).

-ipath *fpattern*

(GNU) Like *-path* (p. ?), but case-insensitive when evaluating the pattern match.

-iregex *ere*

(GNU) Like *-regex* (p. ?), but case-insensitive when evaluating the regular expression match.

-level *n*

(QNX Neutrino) True when the level down in a directory tree is *n*. A file or directory specified on the command line is considered level 0. For example, this command:

```
find /usr -level 1 -type d \
          -print -o \
          -level 2 -prune -type f \
          -name .usrinit -ls
```

displays all the directories in `/usr`, and for each directory that has a `.usrinit` file, displays information on that file in `ls -l` format. (The *-prune* (p. ?) at level 2 prevents unnecessary processing in walking down the directory tree. Though no files further down could possibly match the -level 1 or -level 2 criteria, `find` doesn't detect this automatically — the

Copyright © 2014, QNX Software Systems Limited

command-line expression is applied against every file in the directory tree unless a full recursion of that tree is prevented by a -prune primitive.)

The following command:

```
find /usr -level 1 -ls -prune
```

displays information in `ls -l` format only on files in `/usr` and doesn't descend into any subdirectories of `/usr`.

-links *n*

(POSIX) True if the file has *n* links.

-lname *fpattern*

(GNU) True if *-follow* (p. ?) or *-logical* (p. ?) isn't specified, and the file being evaluated is a symbolic link whose target is a pathname that matches the pattern *fpattern*.

-logical

(QNX Neutrino) Synonym for *-follow* (p. ?).

-ls

(GNU) Similar to *-print* (p. ?), but displays in the same format as *ls* (p. 1139) -l.

Specifying -ls inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-maxdepth *n*

(GNU) Always true. When this flag is set, `find` descends at most *n* levels in the directory hierarchy. Files that are named on the command line are level 0. Note: the + and - modifiers have no meaning when used in conjunction with *n* in this primary.

-mindepth *n*

(GNU) Always true. When this flag is set, `find` doesn't apply the expression to files unless the files are at least *n* levels down in the directory hierarchy. Files named on the command line are level 0. Note: the + and - modifiers have no meaning when used in conjunction with *n* in this primary.

-mmin *n*

(GNU) True if the file data was last modified *n* minutes ago.

-mnewer *file*

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino) True if the file being evaluated was modified more recently than *file* was.

-mount

(GNU) Synonym for -xdev.

-mtime *n*

(POSIX) True if the file modification time subtracted from the time that the `find` utility started running is between *n*-1 and *n* multiples of 24 hours.

-name *pattern*

(POSIX) True if the basename of the filename being examined matches *pattern*. This follows the same pattern-matching rules as used by *fnmatch()* (see the QNX Neutrino *C Library Reference*).

-newer *file*

(POSIX) True if the current file has been modified more recently than *file* has. Note that *file* is evaluated only once.

-nogroup

(POSIX) True if the file belongs to a group ID that isn't in the group database.

-NOP

(QNX Neutrino) Always true, does nothing. This primitive has the side effect of disabling the implicit *-print* (p. ?) that occurs when the expression as a whole evaluates to true. You can use this primitive to benchmark the time it takes to do a walk of the filesystem. For example:

```
find / -NOP
```

-nouser

(POSIX) True if the file belongs to a userid that isn't in the password database.

-ok *utility_name* [*argument...*] ;

(POSIX) Similar to *-exec* (p. ?), except that `find` requests affirmation of the execution of *utility_name* using the current file as an argument by writing to standard error. If the response on standard input is affirmative, the utility is executed. If the response isn't affirmative, the command isn't executed and the value of the -ok operand is false.

Specifying -ok inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

Copyright © 2014, QNX Software Systems Limited

-path *fpattern*

(GNU) True for any file whose path (as would be printed by -print) matches *fpattern*.

-perm [-]*mode*

(POSIX) The *mode* argument represents file mode bits. It's identical to the *symbolic_mode* operand described in *chmod* (p. 124) and is interpreted as follows. To start, a template is assumed with all file mode bits cleared. An *op* symbol of:

Operator	Action
+	Sets the appropriate mode bits
-	Clears the appropriate mode bits
=	Sets the appropriate mode bits, regardless of the process's file mode creation mask

An *op* symbol of - can't be the first character of *mode*.

If the optional hyphen preceding *mode* is omitted, the primary is true when the file permission bits exactly match the value of the resulting template. In addition, the bits associated with the *perm* symbol s are ignored. If the hyphen is included, the primary is true if at least all the bits in the resulting template are set.

-perm [-] *onum*

(POSIX) If the optional hyphen is omitted, the primary is true when the file permission bits exactly match the value of the octal number *onum* and only the bits corresponding to the octal mask 777 are compared.

If the hyphen is included, more flag bits, corresponding to the octal mask 06777, are compared and the primary is true if at least all the bits in *onum* are set.

-pname *pattern*

(QNX Neutrino) Synonym for *-path* (p. ?).

-print

(POSIX) Always true; causes the current pathname to be written to standard output, one pathname per line.

Copyright © 2014, QNX Software Systems Limited

Specifying -print inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-print0

(GNU) Always true. Writes the path currently being evaluated followed by an ASCII NUL character to the standard output.

Specifying -print0 inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-printf ***format***

(GNU) Always true. Write data pertaining to the file currently being evaluated to the standard output, according to the *format* specified. For more information, see "*Formatted printing* (p. 765)," below.

Specifying -printf inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-prune

(POSIX) Stops `find`'s descent from that point in the file hierarchy.

-regex ***ere***

(GNU) True when the pathname of the file currently being evaluated matches the extended regular expression specified by *ere*. See the *grep* (p. 914) documentation for details on regular expressions.

-remove!

(QNX Neutrino) Removes the file being evaluated. If the file is a directory, *rmdir()* is performed, otherwise an attempt is made to *unlink()* the file.

If a directory isn't empty, the attempt to remove it fails. Thus, to recursively remove a directory tree with `find`, the *-depth* (p. ?) primitive must be used in conjunction with -remove!. (Note the simple removal of a directory tree is better and more portably done by using the *rm* (p. 1673) utility.)

This primitive evaluates to TRUE if the removal was successful. Note that the exclamation mark (`!`) is a required part of this primitive.

Specifying -remove! inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-rename ***filestring***

(QNX Neutrino) Renames the file to the pathname indicated by *filestring*. As with other *filestring* arguments, braces (`{}`) encountered in the string are expanded to the name of the file currently being evaluated. A file may

Copyright © 2014, QNX Software Systems Limited

be renamed anywhere within the same filesystem. If the new path lies on another device, the rename fails. Evaluates to true if the rename succeeds, false if it fails.

Specifying -rename inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-size *n*[c]

(POSIX) True if the file size in bytes, divided by 512 and rounded up to the next integer, is *n*. If *n* is followed by c, the size is in bytes.

-spawn *cmd* [*arguments*]... ;

(QNX Neutrino) Similar to *-exec* (p. ?), except that the command is invoked directly (i.e. not through a shell). The -spawn primary is faster but less flexible than -exec.

Specifying -spawn inhibits the automatic *-print* (p. ?) when the expression as a whole evaluates to true.

-true

(GNU) Always true.

-type *c*

(POSIX) True if the file type is *c*, where *c* is one of:

Filetype	Description
b	Block special file
c	Character special file
d	Directory
p	FIFO
f	Regular file
l	Symbolic link
n	Named special file

-uid *n*\|*userid*

(GNU) Synonym for *-user* (p. ?).

-used *n*

(GNU) True if file was last accessed *n* days after its status was last changed.

Copyright © 2014, QNX Software Systems Limited

-user *uname*

(POSIX) True if the file belongs to the user *uname*. If *uname* is numeric and the name doesn't exist in the password database, *uname* is taken as a user ID. Note that *uname* is evaluated only once.

-xdev

(POSIX) Always true — stops `find` from descending past directories that have a different device ID. If any -xdev primary is specified, it applies to the entire expression, even if the -xdev primary isn't normally evaluated.

Formatted printing (-printf and -fprintf primitives)

The *-printf* (p. ?) and *-fprintf* (p. ?) primaries require as arguments a format string similar in appearance to that used in the C language *printf()* function. The format string consists of regular ASCII characters and a set of special codes starting with percent (%) format codes and backslash (\) escape codes.

Backslash (\) Escape Codes

****

Literal backslash (\) character.

\a

Alarm bell.

\b

Backspace character.

\c

Stop printing from format and flush output.

\f

Form-feed character.

\n

Newline character.

\r

Carriage return character.

\v

Vertical tab character.

Format Codes

Copyright © 2014, QNX Software Systems Limited

%%

Literal percent (%) character.

%p

The pathname currently being evaluated.

%f

The basename of the file.

%h

The dirname of the file.

%P

The name of the file with the root of the file tree (pathname specified on the command line) removed from the beginning.

%H

The name of the root of the file tree (pathname specified on the command line under which the current file was found).

%g

The file's group name, or numeric group ID if no name found.

%G

The file's numeric group ID.

%u

The file's user name, or numeric userid if no name found.

%U

The file's numeric userid.

%m

The file's permissions in octal.

%k

The file's size in 1k blocks (rounded up).

%b

The file's size in 512-byte blocks (rounded up).

%s

Copyright © 2014, QNX Software Systems Limited

The file's size in bytes.

%d

The depth of the file in the directory tree. The files specified on the command line have a depth of 0.

%l

If the file is a symbolic link, the filename of the link object. Null if file isn't a symbolic link.

%i

The inode number of the file.

%n

The link count of the file.

%a

The file's last access date and time; equivalent to %Ac

%A*fchar*

Partial or full representation of the file's last access time, depending on the format character *fchar*:

fchar	Is replaced by:
a	Abbreviated weekday name
A	Full weekday name
b	Abbreviated month name
B	Full month name
c	Locale's appropriate date and time representation
d	Day of the month as a decimal number (01-31)
D	Date in the format *mm*/*dd*/*yy*
h	Abbreviated month name
H	Hour (24 hr) as a decimal number (00-23)
I	Hour (12 hr) as a decimal number (01-12)

Copyright © 2014, QNX Software Systems Limited

fchar	Is replaced by:
j	Day of the year as a decimal number (001-366)
m	Month as a decimal number (01-12)
M	Minute as a decimal number
n	Newline character
p	AM or PM
r	12-hr clock time (01-12) using the AM/PM notation i.e. *hh*:*mm*:*ss* (AM|PM)
S	Second as a decimal number (00-59)
t	Tab character
T	24-hr clock time (00-23) *hh*:*mm*:*ss*
U	Week number of the year as a decimal number (00-52), where Sunday is the first day of the week
w	Weekday as a decimal number (0-6), where 0 is Sunday
W	Week number of the year as a decimal number (00-52), where Monday is the first day of the week
x	Locale's appropriate date representation
X	Locale's appropriate time representation
y	Year without century as a decimal number
Y	Year with century as a decimal number
Z	Timezone name, NULL if no timezone exists

QNX Neutrino currently supports only the POSIX (i.e. C) locale.

Copyright © 2014, QNX Software Systems Limited

%c

File's last status change date and time — equivalent to %Cc

%C*fchar*

Partial or full representation of the file's last status change time, depending on the format character *fchar*. See description of %A, above for details.

%t

File's last modification date and time — equivalent to %Tc

%T*fchar*

Partial or full representation of the file's last modification time, depending on the format character *fchar*. See description of %A, above for details.

Examples:

Search the filesystem for the `myfile` file or directory:

```
find / -name myfile
```

Remove all files named `tmp` or ending in `.xx` that haven't been accessed for seven or more 24-hour periods:

```
find / \( -name tmp -o -name '*.xx' \) \
     -atime +7 -exec rm {} \;
```

Print the pathnames of all files in and below the current directory, but skip any directories named SCCS and the files beneath them:

```
find . -name SCCS -prune -o -print
```

Note that when possible, it's better to use `find` in combination with *xargs* (p. 2072) than it is to use the *-exec* (p. ?) or *-spawn* (p. ?) options to start commands. You can use `xargs` to start a program once for many files; -exec or -spawn starts the program once for every file matched. You'll see a tremendous difference in speed between the two approaches. For instance:

```
find / -name '*.tmp' | xargs rm
```

is generally preferable to:

```
find / -name '*.tmp' -exec rm {} \;
```

See `xargs` for more details.

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

All path operands were successfully traversed.

>0

An error occurred.

Caveats:

If you use *-exec* (p. ?), *-ok* (p. ?), or *-spawn* (p. ?), `find` catches `SIGINT` (which you can generate, for example, by pressing **Ctrl**–**C** or **Ctrl**–**Break**) and asks you whether the find should continue. If these primaries aren't used, `find` terminates from `SIGINT`.

Copyright © 2014, QNX Software Systems Limited

flashctl

Manage a flash filesystem

Syntax:

```
flashctl [-eFfimruvxz] [-A] [-a flags] [-b align]
         [-c flag] [-L] [-l limit] [-n path] [-o offset]
         -p path [-s num] [-U] [-u] [-v...]
```

Runs on:

QNX Neutrino

Options:

The options are processed in the order given, and apply to the last partition specified by a -p option, letting you control multiple partitions with one command.

-A

Unlock the entire flash array specified by the preceding -p option.

Use this option only with a "raw" device (e.g. `/dev/fs0`); don't use it with a mounted partition. You can't use the -l and -o options with the -A option.

-a *flags*

The mount-attribute flags (the `_MOUNT_*` bits defined in `<sys/mount.h>`, and the `IOFUNC_MOUNT_*` bits defined in `<sys/iofunc.h>`). The default is 0.

-b *align*

Set the alignment to 2^{align}. The default value of *align* is 2, resulting in an alignment of 2^2.

If you start the *devf-** (p. 311) driver with the -D (automatic detection of error-correcting code mode) or -x (enable software ECC mode) option, you need to specify the alignment. If you're using 64-byte alignment ECC, specify 6. For example:

```
devf-generic -x1 -s0x20000000,16M
flashctl -p /dev/fs0 -o1M -ev
flashctl -p /dev/fs0p0 -o1M -f -b6
```

Copyright © 2014, QNX Software Systems Limited

If you're using 32-byte alignment ECC, specify 5 for *align*. For example:

```
devf-generic -x2 -s0x20000000,16M
flashctl -p /dev/fs0 -o1M -ev
flashctl -p /dev/fs0p0 -o1M -fv -b5
```

-c *flag*

Set the compression flag (default 1).

-e

Erase the raw partition specified by the preceding -p option from the *offset* specified by the -o option through to *limit* specified by the -l option.

If you specify the -v option for verbose output, `flashctl` outputs a period (`.`) for every partition unit that it erases. On RAM disk, the unit size is 64K; on other devices, it varies. This output is reassuring because it can take several minutes to erase slow flash devices.

-f

Format the raw partition specified by the preceding -p option from the *offset* specified by the -o option through to *limit* specified by the -l option.

-F

Force the driver to use the given offset and length parameters, even if the offset and length parameters don't fall within the bounds of the partition specified by the preceding -p option. The range provided by the -o and -l arguments must fit in a single partition, and can't cross partition boundaries.

-i

Print filesystem information for the partition specified by the preceding -p option; see "*Filesystem information* (p. 774)", below.

-L

Lock the raw partition specified by the preceding -p option from the *offset* specified by the -o option through to *limit* specified by the -1 option.

The -L option fails if the partition (e.g. `/dev/fs0p0`) is mounted.

You can use `/dev/fs0` to bypass the mounted filesystem check, but we don't recommend that you do so, because the `/dev/fs0` path bypasses all protection mechanisms. If you use it, the flash driver doesn't know what filesystem or partition the locking affects. The filesystem might think the flash is writable and suddenly it isn't. The filesystem will get very confused, and write operations will start

Copyright © 2014, QNX Software Systems Limited

to fail. Worse yet, if you lock a single block in a filesystem via `/dev/fs0`, *most* of your writes will fail and suddenly stop working.

-l *limit*

Specify a size limit in bytes (default 2G) for the partition specified by the preceding -p option. The *limit* can include a suffix of `k`, `K`, `m`, `M`, `g`, or `G`.

-m

Mount the filesystem partition specified by the preceding -p option. The -m option doesn't work when you use the -o offset option. You must restart the driver for the partition to be recognized.

-n *path*

Specify the filesystem mountpoint for the partition specified by the preceding -p option.

This option overrides any mountpoint specified with the `mount` attribute of the *`mkefs`* (p. 1209) command.

-o *offset*

Specify the offset in bytes (default 0 bytes). The *offset* can include a suffix of `k`, `K`, `m`, `M`, `g`, or `G`.

-p *path*

Specify the raw mountpoint for the partition.

-r

Reclaim deleted blocks on the partition specified by the preceding -p option, up to the *limit* specified by the -l option.

-s *num*

Specify the number of spare blocks (default `1`) on the partition specified by the preceding -p option.

-U

Unlock the raw partition specified by the preceding -p option from the offset specified by the -o option through to limit specified by -l option.

The -U option fails if the partition (e.g. `/dev/fs0p0`) is mounted.

You can use `/dev/fs0` to bypass the mounted filesystem check, but we don't recommend that you do so.

Copyright © 2014, QNX Software Systems Limited

-u

Unmount the filesystem partition specified by the preceding -p option.

-v...

Be verbose; more `v` characters cause more verbosity.

-x

Exit the driver.

-z

Query for the compression flag.

Description:

The `flashctl` utility is used to manage a flash filesystem. The utility interacts with the flash filesystem driver using *devctl()* messages. Using `flashctl`, you can erase and format a raw partition, force a reclaim operation, and get information about flash filesystem partitions.

- The options are processed in the order entered. For example, you must erase (-e), format (-f), and mount (-m) a partition in this order.
- For now, use `flashctl` instead of `mount` and `umount` to mount and unmount flash partitions.

The `flashctl` utility rounds the values of the -o and -l ("el") options down to the nearest block bound. If the range specified exceeds the partition size, it's rounded down to fit. If you use the -v option, `flashctl` displays what the values have been rounded to.

Filesystem information

If you use the -i option, `flashctl` displays information about the partition that you chose with the -p option; the amount of information depends on what you chose (the raw socket, raw partition, or formatted mounted filesystem):

- `/dev/fs0` points to the entire socket, which means the whole chip (or all physically contiguous chips). This is the `Array Info` listed below.
- `/dev/fs0p`*X* refers to a partition, referred to as a `Part`.
- The erase sectors on flash are referred to as `Units`.

Here's an example of the output:

```
Array Info
 Total      :  0x00800000  100%
  Chip Size :  0x00800000  100%
  Unit Size :  0x00020000    1%
Part Info
```


Copyright © 2014, QNX Software Systems Limited

```
 Total      :  0x00800000  100%
  Spare     :  0x00020000    1%
  Headers   :  0x00001A34    0%
  Padding   :  0x00000000    0%
   Overhead :  0x00021A34    1%
  Free      :  0x007DE5AC   98%
  Stale     :  0x00000000    0%
   Avail.   :  0x007DE5AC   98%
  Reserved  :  0x00000020    0%
Unit Info
 Erase Stats
  Average   :  0
  Minimum   :  0
  Maximum   :  0
  Total     :  0
```

Here's how to interpret the output:

`Array Info`

The total amount of contiguous flash at the specified base address:

- `Total` — the size.
- `Chip Size` — the size of each chip if there are multiple physical chips (they're assumed to all be the same size).
- `Unit Size` — the size of one sector, and the percentage of the total size.

`Part Info`

Information for the requested partition. Sizes are given in bytes and as a percentage of the size of the partition; some fields might not be filled in:

- `Total` — the partition size.
- `Spare` — the amount of space used for spare blocks; `Spare`/`Unit Size` gives the number of spare blocks.
- `Headers` — the space used for filesystem extent headers.
- `Padding` — the space used for alignment padding.
- `Overhead` — the total of `Headers` + `Padding`.
- `Free` — the amount of free space (no reclaim necessary).
- `Stale` — the amount of stale space.
- `Avail.` — the total amount of usable space; this is the total of `Free` + `Stale`.
- `Reserved` — internal reserved space for unlinking, etc.

`Unit Info`

Information pertaining to individual sectors:

- `Erase Stats` — the average, minimum, maximum, and total number of erasures performed within the specified partition.

Copyright © 2014, QNX Software Systems Limited

Examples:

Create a flash filesystem between 1 MB and 3 MB in a 4 MB raw partition:

```
flashctl -p /dev/fs0p0 -o 1M -l 2M -e -f
```

This command line results in this organization:

`/dev/fs0p0`

A raw partition at 0–1 MB

`/dev/fs0p1`

A flash filesystem partition at 1–3 MB (mounted as `/fs0p1`)

`/dev/fs0p2`

A raw partition at 3–4 MB

Erase, format, and then mount (as `/fs0p0`) the flash filesystem partition `/dev/fs0p0`:

```
flashctl -p /dev/fs0p0 -e -f -m
```

Mount the given flash filesystem partition as `/flash`:

```
flashctl -p /dev/fs0p0 -n /flash -m
```

Use the following command to format the range from 3 to 6 MB, if the layout is unknown:

```
flashctl -p /dev/fs0p0 -F -o 3m -l 6m -f
```

For *devf-ram* (p. 320) partitions only, you must format and erase a partition before you can mount the flash filesystem. Otherwise, you may get an error message `flashctl: mounting partition failed.`

Caveats:

The mount facility (with the -m and -u options) doesn't provide complete `mount` functionality, such as read-only (`mount` -r) and special options (`mount` -o).

Copyright © 2014, QNX Software Systems Limited

flex

Generate programs for lexical tasks

Syntax:

```
flex [-bcdfhilnpstvwBFILTV78+? -C[aefFmr] -ooutput
      -Pprefix -Sskeleton] [--help --version] [file ...]
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-+

Generate C++ scanner class.

-7

Generate 7-bit scanner.

-8

Generate 8-bit scanner.

-?

Display a help message.

-B

Generate batch scanner (opposite of -I).

-b

Generate backing-up information to `lex.backup`.

-C

Specify the degree of table compression (default is -Cem):

- -Ca — trade off larger tables for better memory alignment.
- -Ce — construct equivalence classes.
- -Cf — don't compress scanner tables; use -f representation.
- -CF — don't compress scanner tables; use -F representation.
- -Cm — construct meta-equivalence classes.
- -Cr — use *read()* instead of standard I/O for scanner input.

Copyright © 2014, QNX Software Systems Limited

-c

Do-nothing POSIX option.

-d

Turn on debug mode in generated scanner.

-F

Use the alternative fast scanner representation.

-f

Generate fast, large scanner.

-h

Display a help message.

-I

Generate an interactive scanner (opposite of -B).

-i

Generate case-insensitive scanner.

-L

Suppress #`line` directives in scanner.

-l

("el") Provide maximal compatibility with original `lex`.

-n

Do-nothing POSIX option.

-o*output*

Specify the output filename.

-P*prefix*

Specify a scanner prefix other than `yy`.

-p

Generate a performance report to *stderr*.

-S*skeleton*

Specify the skeleton file.

-s

Copyright © 2014, QNX Software Systems Limited

Suppress default rule to ECHO unmatched text.

-T

Run `flex` in trace mode.

-t

Write the generated scanner on *stdout* instead of `lex.yy.c`.

-V

Report the `flex` version.

-v

Write a summary of scanner statistics.

-w

Don't generate warnings.

--help

Display a help message.

--version

Report the `flex` version.

Description:

The `flex` command generates programs for lexical tasks.

We don't provide the flex library (`libfl.a`). If you wish to generate code that is not dependent on this library, add the following line as the first line in your lex source file:

```
%option noyywrap
```

Otherwise, the undefined *yywrap()* function will result in a link error.

For more information, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

fmt

Format concatenated input

Syntax:

```
fmt [-w width] [-m maximum] [-] [file...]
```

Runs on:

QNX Neutrino

Options:

-m *maximum*

The maximum line length. The default is 75 characters.

-w *width*

Make the lines as close to *width* as possible. The default width is 65 characters.

-

Read from standard input

***file*...**

The file or files to be formatted.

Description:

The `fmt` utility concatenates the input files, formats the output as specified by the width and maximum options, and sends the results to standard output. Non-indenting tabs are converted to spaces; tabs are used for indentation where possible. If no input files are given, `fmt` reads from standard input.

Copyright © 2014, QNX Software Systems Limited

fold

Fold lines (POSIX)

Syntax:

```
fold [-bs] [-w width] [file...]
```

Runs on:

QNX Neutrino

Options:

-b

Count width in bytes rather than column positions. If -b is specified, `carriage-return`, `backspace` and `tab` are treated as normal characters. Only `newline` remains special.

-s

If a segment of a line contains a blank within the first *width* column positions, break the line after the last such blank meeting the width constraints (i.e. avoid breaking the line in the middle of a word).

-w *width*

Specify the maximum line length. The default is 80 characters.

Description:

The `fold` filter folds lines in files by breaking up lines that have a width in excess of 80 or, if specified, the *width* in the command-line -w option. Lines are folded by the insertion of a `newline` character. All output is written to the standard output.

In all cases, the current count of line width is reset to zero when a `newline` character is encountered in the input. In addition, when the -b option is not specified, the following characters have a special meaning when encountered in the input:

`carriage-return`

The current count of line width is set to zero.

`backspace`

The current count of line width is decremented by one (but it doesn't go below zero).

Copyright © 2014, QNX Software Systems Limited

tab

The current count of line width is incremented to the next count for which *count* modulo 8 equals 1.

Examples:

Fold the file `myfile.txt` to a maximum line width of 80 characters, making the line breaks on the last word boundary (white space) before the width was exceeded and write the results to the standard output:

```
fold -s myfile.txt
```

Fold the file `myfile.txt` to a width of 40 characters, and treat all characters except `newline` as occupying one character (byte) position and write the results to the standard output:

```
fold -bw40 myfile.txt
```

Files:

If no *file*s are specified on the command line, `fold` reads lines to be folded from the standard input until `EOF` is reached.

The `fold` utility writes all folded output from all input files to the standard output.

If an error occurs, a diagnostic message is written to the standard error.

The `fold` utility reads lines from the text files specified on the command line. If the -b is specified, the input files need not be text files.

Exit status:

>0

An error occurred.

0

All files were processed successfully.

Copyright © 2014, QNX Software Systems Limited

fpemu.so

Provide support for floating-point emulation

Syntax:

N/A

Runs on:

QNX Neutrino

Targets:

x86

Options:

None.

Description:

The fpemu.so shared object provides support for floating-point emulation. The QNX Neutrino RTOS automatically loads this object when math operations are required on a system with no floating-point unit.

By default, QNX Neutrino uses floating-point hardware in the CPU if present, and emulates it in software if the CPU has no FPU. To force floating-point emulation, use the -fe option to *procnto* (p. 1586).

Caveats:

In this version of QNX Neutrino, you can't use the floating point emulator in statically linked executables.

Copyright © 2014, QNX Software Systems Limited

freeze

Compress and uncompress files (UNIX)

Syntax:

```
freeze [-cdfvVz] [[+n1,...,n8] [filename]]...
```

Runs on:

QNX Neutrino

Options:

-c

Write the results of the freeze/melt operation to standard output. No files are changed.

-d

Uncompress the files from the archive (`melt`).

-f

Force creation of the `.F` file. This option creates the `.F` file, even if one already existed, without prompting you for confirmation. The `.F` file is created even if the compressed file is larger than the original.

-V

Display the program's version number and compilation options.

-v

Display a counter. For each file being frozen, display a kilobytes counter, and, after the file is frozen, print a message giving the percentage of the input file's size that has been saved by compression.

-z

If the output of a melt (option -d) is to a tty, allocate a larger output buffer so screen output is in fullscreen chunks.

+n1,...,n8

A list of 8 numbers separated by commas, specifying the values for the static Huffman table.

Copyright © 2014, QNX Software Systems Limited

Description:

The `freeze` utility compresses the specified files or the standard input. If a file becomes smaller, it's replaced by a file with the extension `.F` (you can use the -f option to force the creation of the `.F` file, even if the compressed file is larger). If no files are specified, the compression is applied to the standard input and is written to the standard output.

If you don't specify the -f option, and you run `freeze` in the foreground, you're prompted as to whether the file should be overwritten.

When a `.F` file is created, the original file is removed.

Normally there are several links set up to the `freeze` utility. Freeze behaves differently when invoked under these command names:

This link:	Is equivalent to:
melt (p. 1197)	`freeze` -d
fcat (p. 729)	`freeze` -cd

You can restore compressed files to their original form by:

- Specifying the -d option to `freeze`.

  Or

- Running *melt* (p. 1197) on the `.F` files or on the standard input.

When you specify filenames, `freeze` maintains the ownership, modes, access times, and modification times between the file and its `.F` version. As a result, you can use `freeze` for archival purposes, yet you can still use it with *make* (p. 1166) after melting.

The `freeze` utility uses the Lempel-Ziv algorithm on the first pass and the dynamic Huffman algorithm on the second pass. The size of “sliding window” is 8K and the maximum length of “matched string” is 256. The positions on the window are coded using a static Huffman table.

A two-byte magic number is prepended to the file to ensure that neither melting of random text nor refreezing of already frozen text is attempted. In addition, the characteristics of the static Huffman table being used during the freeze are written to the file so that these characteristics may be adapted to concrete conditions.

The amount of compression you obtain depends on the size of the input file and the distribution of character substrings and their probabilities. Typically, text files (e.g. C programs) are reduced by 60% to 75%, while executable files are reduced by 50%. Compression is generally much better than that achieved by LZW coding or by Huffman coding, although it takes more time to compute.

Copyright © 2014, QNX Software Systems Limited

An argument preceded by a + defines the values to use for the Huffman table. You may want to explicitly state these values in order to modify the compression algorithm.

The `freeze` compression utility will eventually become deprecated in favor of the GNU `zip` suite, *`gzip`* (p. 921)/*`gunzip`* (p. 920)/*`zcat`* (p. 2080). The `freeze` suite of utilities will continue to be provided for quite some time before being eliminated completely.

Examples:

Freeze all files in the current directory:

```
freeze *
```

Extract all C source and header files:

```
melt *.[ch].F
```

Or:

```
freeze -d *.[ch].F
```

View the concatenated contents of all compressed files in the current directory:

```
fcat *.F | more
```

Exit status:

0

Normal exit.

1

An error occurred.

2

The last file grew after freezing.

Contributing author:

Leonid A. Broukhis

Copyright © 2014, QNX Software Systems Limited

fs-cd.so

ISO-9660 filesystem support (with extensions)

Syntax:

```
driver ... cd cd_options ... &
```

Runs on:

QNX Neutrino

Options:

Where *driver* is one of the `devb-*` drivers, and *cd_options* is one or more of the following, separated by commas:

case=asis|lower|upper

Control the case used to display ISO-9660 filenames (to a *readdir()* request; all pathname matching is always performed case-insensitively on such names):

- `asis` — don't convert the filename in any way; if the CD was mastered with strict ISO-9660 compliance, the name will be in uppercase, but more lenient utilities could produce mixed-case filenames.
- `lower` — convert to lowercase (the default).
- `upper` — convert to uppercase.

RRIP and Joliet store case-preserving names and ignore this option.

exe

Set execute permission (on all non-RRIP regular files).

gid=*group*

Set the owning group to *group* for files on disks without Rock Ridge extensions. The default is `0` (`root`).

hidden=*hidden_mode*

Specify what to do with "hidden" files. The *hidden_mode* can be one of:

- `ignore` — ignore the hidden files; they don't appear in the filesystem.
- `show` — (the default) show hidden files in the filesystem as normal files.

Copyright © 2014, QNX Software Systems Limited

- `dot` — show hidden files in the filesystem with a dot (`.`) prefixed to their names.

info=*path*

The name of the information pseudo-directory (the default is `-.info.`).

If the option is empty, no metadata directory is created. If the option begins with a `-`, then blank metadata fields are hidden; if the option begins with a `+`, then all metadata fields are presented but may have a length of zero.

noaudio

Disable audio extensions (the `.info./audio` control file).

nohsf

Disable High Sierra format.

nojoliet

Disable Microsoft Joliet extensions.

nomulti

Disable multisession support (mount the first data session instead of the last).

norrip

Disable Rock Ridge extensions.

uid=*user*

Set the owner to *user* for files on disks without Rock Ridge extensions. The default is `0` (`root`).

umask=*mask*

Apply this permission mask to files on disks without Rock Ridge extensions. The default is 0 (all access permissions).

In addition, you can specify any of the *filesystem options* (p. 1000) described for *`io-blk.so`* (p. 993).

Description:

The `fs-cd.so` shared object provides support for ISO-9660 filesystems. It's automatically loaded by the `devb-*` drivers when mounting an ISO-9660 filesystem.

Copyright © 2014, QNX Software Systems Limited

- This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).
- We've deprecated `fs-cd.so` in favor of *`fs-udf.so`* (p. 829), which now supports ISO-9660 filesystems in addition to UDF.

Copyright © 2014, QNX Software Systems Limited

fs-cifs

Common Internet Filesystem or SMB client filesystem (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
fs-cifs [-a] [-b] [-D] [-d name] [-h] [-L|-l] [-o
option[,option...]]
        [-t n] [-v[v]...] [-Z n]
        [[//netbiosname:]server:/share
        prefix user passwd]
```

Runs on:

QNX Neutrino

Options:

-a

Spoof POSIX attributes; calls to *chmod()* and *chown()* return EOK instead of ENOTSUP. Use -a to get rid of error messages from applications that attempt to set the mode or ownership of files and directories, such as *cp* (p. 141).

-b

Turn off buffered writes (default is on). This makes writes to the remote filesystem slower, but safer.

-D

Run in the foreground.

-d *name*

Specify the domain name.

-h

Display usage information.

-L

Ask the user to provide the password (the command line doesn't include *password*).

Copyright © 2014, QNX Software Systems Limited

-l

("el") Ask the user to provide the name of the user and the password (the command line doesn't include *user* or *password*).

-o *option*[,*option*...]

Miscellaneous options; use commas to separate them. These options include:

- old — don't use 64-bit filesystem dialects. By default, `fs-cifs` supports 64-bit filesystem operations.
- plainpwd — if logging in with an encrypted password fails, try to log in by using the password unencrypted.

Sending passwords in plain text may be considered a security problem.

If `fs-cifs` failed to authenticate with the server using an encrypted password, it used to then attempt to authenticate with the server using an older method while sending the password unencrypted. In QNX Neutrino 6.3.2 and later, `fs-cifs` sends the password encrypted, unless you specify the option -o plainpwd. You might need this option when mounting shares on older versions of Windows.

- showpwd — show the plain-text password in the log file.

Adding the password to the log file may be a security problem if unauthorized personal have access to the log file.

- timeout=*n* — set the reconnection timeout to be *n* seconds.

-t *n*

Allow up to *n* threads (default is 5). Use -t to tune resource usage; more threads require more resources (such as memory), but decrease filesystem latency.

-v[v...]

Verbose output. Additional `v` characters give more verbose output.

-Z *n*

The value of *n* indicates how to attach to the path:

- B or b — attach before other managers.
- A or a — attach after other managers.

Copyright © 2014, QNX Software Systems Limited

The default is neither. For more information, see "Ordering mountpoints" in the Process Manager chapter of the *System Architecture* guide.

[*//netbiosname*:]*server*:*/share*

Specify the server name or IP address and the shared resource name of the filesystem being mounted.

If you specify a server name, there must be some way to resolve it to an IP address. See `/etc/hosts` (p. 946) or `/etc/nsswitch.conf` (p. 1369).

You must specify the *netbiosname* if:

- The server's TCP/IP host name is different from its NetBIOS name

  Or

- The server host is running a Microsoft operating system.

prefix

Absolute path specifying where to mount *share* in the local filesystem.

user

Log on to the SMB server as *user*.

passwd

Log on to the SMB server with *passwd*.

If your password includes characters that the shell considers to be special, you might have to enclose the password in quotation marks.

Description:

The `fs-cifs` filesystem manager is an SMB (also known as CIFS — Common Internet Filesystem) client operating over TCP/IP. SMB is a protocol for accessing resources in a controlled fashion over a LAN.

The `fs-cifs` filesystem manager is primarily intended for use as a client with Windows NT machines, although it also works with any SMB server (such as OS/2 Peer, LAN Manager, or SAMBA). To use `fs-cifs`, you must have an SMB server and a valid login on that server.

The `fs-cifs` filesystem manager also requires a TCP/IP transport layer, such as the one provided by `io-pkt*` (p. 1007).

Copyright © 2014, QNX Software Systems Limited

You can mount SMB filesystems at the same time as you start `fs-cifs`; you can also mount them separately after you have started `fs-cifs` using the *mount* (p. 1312) command.

If you want to mount filesystems separately, start `fs-cifs` without arguments, as a daemon, then create your mountpoints using `mount` specifying `cifs` as the *type*. See below for an example.

If *syslogd* (p. 1885) is running, `fs-cifs` writes any error messages to the system log.

Examples:

Start `fs-cifs` and mount the `QNX_BIN` share as `/bin` from an SMB server named SMB_SERVER (with IP address 10.0.0.1) using the `guest` account and a password of `none`:

```
fs-cifs SMB_SERVER:/QNX_BIN /bin guest none
```

Or:

```
fs-cifs 10.0.0.1:/QNX_BIN /bin guest none
```

The same as the above, but with a server, called NB_NAME, running a Microsoft operating system:

```
fs-cifs //NB_NAME:SMB_SERVER:/QNX_BIN /bin guest \
none
```

Or:

```
fs-cifs //NB_NAME:10.0.0.1:/QNX_BIN /bin guest \
none
```

Ask the user for the password:

```
fs-cifs -L //NB_NAME:SMB_SERVER:/QNX_BIN /bin guest
```

Ask the user for the name of the user and the password:

```
fs-cifs -l //NB_NAME:SMB_SERVER:/QNX_BIN /bin
```

Mounts the server as the user in the QNX domain:

```
fs-cifs -d QNX //MS:10.1:/QNX_BIN /mnt user passwd
```

Start `fs-cifs` as a daemon, then mount the `QNX_BIN` share as `/bin` from an SMB server named SMB_SERVER (with IP address 10.0.0.1) using the `guest` account and a password of `none`:

```
fs-cifs &
```

Then:

```
mount -t cifs -o guest,none //SMB_SERVER:10.0.0.1:/QNX_BIN /bin
```

or:

```
mount -t cifs -o user=guest,password=none \
//SMB_SERVER:10.0.0.1:/QNX_BIN /bin
```

Copyright © 2014, QNX Software Systems Limited

Files:

`/dev/log`

syslogd (p. 1885) interface.

Caveats:

When the filesystem is mounted, everybody who uses the filesystem does so with the privileges of the *user* specified on the command line.

The *passwd* argument is required on the command-line even if *user* doesn't require a password; it can be anything other than whitespace.

Copyright © 2014, QNX Software Systems Limited

fs-dos.so

DOS filesystem (QNX Neutrino)

Syntax:

```
driver ... dos dos_options ... &
```

Runs on:

QNX Neutrino

Options:

Where *driver* is one of the `devb-*` drivers, and *dos_options* is one or more of the following, separated by commas:

case

Use case-sensitive filename matching (forces long filenames). DOS/FAT is normally a case-preserving, case-insensitive filesystem.

codepage=*mapping*

Install a DOS codepage for mapping of locale 8.3 filenames. These names are used only when the corresponding (Unicode) long filename is absent (created pre-Win95) or has been disabled (using lfn=ignore) or for the volume label; specification of the appropriate locale will also allow portability of filenames created by `fs-dos.so` to older versions of DOS. Use the "chcp" native command on a DOS system to determine its active codepage. Supported values for *mapping* are: `cp437`, `cp850`, `cp852`, `cp866`, `cp1250`, `cp1251`, `cp1252`, and `koi8r`.

compat=*mode*

Set DOS/Windows compatibility mode. Certain versions of DOS implement minor individual peculiarities of the FAT on-disk format, although this is unlikely to affect any normal or typical filesystem client usage. Supported values for *mode* are: `dos`, `os2`, `win95`, `win98`, `win2k`, and `auto` (the default).

exe=*exec_mode*

Specify how to handle the POSIX x-bit for executables; *exec_mode* can be one of:

all

Copyright © 2014, QNX Software Systems Limited

Make all files executable.

none

Make no files executable.

system

Use the DOS "system" attribute to indicate which files are executable.

auto

Make files ending in `.exe`, `.bat`, and `.com` executable.

The default is `auto`.

fat=lazy|nonrmv|always

Set pre-reading of the FAT. Scanning the FAT is required to return the count of free blocks. It also allows for improved *write()* performance by creating an in-memory summary of where free blocks might be located within the filesystem.

The value must be one of the following:

- `always` — always read the entire FAT at mount time
- `lazy` — read the FAT only when required (*statvfs()* query)
- `nonrmv` — act as `always` for nonremovable media, and `lazy` for removable media

The default is `nonrmv`.

fsi=*mode*

Set the handling of the FAT32/FSI record (this contains a count of free clusters plus a hint at the next free cluster). The *mode* is one of:

- `ignore` — ignore the record.
- `lazy` — update the record only when unmounting.
- `update` — update it whenever the FAT is modified.
- `use` — update it and use it (normally the free block count is calculated at mounting).

The default is `lazy`.

gid=*group*

Set the owning group of all files to *group*. The default is `0` (`root`).

Copyright © 2014, QNX Software Systems Limited

hidden=*hidden_mode*

Specify what to do with files that have the DOS "hidden" attribute; *hidden_mode* can be one of:

ignore

The hidden files are ignored; the files don't appear in the filesystem.

show

The hidden files appear in the filesystem as normal files.

dot

The hidden files appear in the filesystem with a dot (`.`) prefixed to their names. Files created with a leading dot have the DOS "hidden" attribute set.

The default is `show`.

lfn=*lfn_mode*

Specify what to do with long filenames:

ignore

Ignore long filenames. Only 8.3 filenames are displayed or created.

show

Show long filenames. Long filenames are created if the filename is longer than 8.3 or if mixed case is used.

always

Always create both short and long filenames.

The default is `show`.

lnk=*lnk_mode*

Set the handling of Windows *shortcut* files. The options are:

ignore

Place no special meaning on these files (the default).

all

Turn all shortcut files into symbolic links pointing to their shortcut targets.

Copyright © 2014, QNX Software Systems Limited

local

Turn into symbolic links only those shortcut files that point to targets within this filesystem.

notrunc

Enforce short (8.3) filenames. This option is valid only with lfn=ignore. By default, filename components beyond the 8.3 limit are silently ignored. For example, `LONGFILENAME.TXT` becomes `LONGFILE.TXT`.

posix=*posix_mode*

Set POSIX check and emulation modes; *posix_mode* can be one of the following:

none

Disable POSIX checks and emulation.

- There are no `.` and `..` entries in the root directory.
- The link count of every directory is always 2.

emulate

Provide the following features beyond those provided by FAT:

- Create the `.` and `..` entries in the root directory.
- Calculate directory size.
- Calculate the number of links per directory, based on the subdirectories.
- Ignore modification attempts that cannot be stored on disk or recreated by emulation, but don't raise errors.

strict

Set stricter POSIX checks. Provide the same features listed under `emulate` mode, but flag as errors modification attempts that cannot be stored on disk or recreated by emulation. E.g. an error of `EINVAL` is flagged for attempts to do any of the following:

- Set the userid or group ID to something other than the default.
- Remove an `r` permission.
- Set an `s` permission.
- Set a file modification or access time to pre-1980.

The default is `emulate`.

Copyright © 2014, QNX Software Systems Limited

sfn=*mode*

Set the display of 8.3 filenames; *mode* is one of:

- `lower` — always user lowercase (`file.c`).
- `upper` — always use uppercase (`FILE.C`).
- `windows` — emulate WindowsNT and use all lowercase or all uppercase according to the attributes of each filename (e.g. `file.c` or `FILE.C`).

The default is `lower`.

uid=*user*

Set the owner of all files to *user*. The default is `0` (`root`).

umask=*mask*

Apply this permission mask to all files. The default is 0 (all permissions).

vollabel=*vollabel_mode*

Specify what to do with the DOS volume name; *vollabel_mode* can be one of:

`ignore`

The volume label isn't displayed.

`show`

The volume label is displayed as a name-special file.

`equals`

The volume label is displayed as a name-special file with an = prefixed to its name.

The default is `equals`.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *`io-blk.so`* (p. 993).

Description:

The `fs-dos.so` shared object lets you mount DOS filesystems (FAT12, FAT16, and FAT32) under QNX Neutrino.

The `fs-dos.so` shared object is automatically loaded by the `devb-*` drivers when mounting a DOS FAT filesystem.

Copyright © 2014, QNX Software Systems Limited

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

fs-etfs-ram

Embedded transaction filesystem for RAM/SRAM

Syntax:

```
fs-etfs-ram [common-options] [-Ddriver-options]
```

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

Board Support Packages may include a board-specific embedded transaction filesystem. All `fs-etfs-*` filesystems use the same common options as `fs-etfs-ram`; for driver-specific options, see the BSP documentation or use the -D use option, as described in "*Driver options* (p. 804)," below.

Common options

-a

Update access times (*atime*). Default is to not update *atime*, to reduce the number of flash writes.

-B

Don't detach and run in the background. This is useful for debugging.

-b *priority*

Run background reclaim at this priority. The default is 8.

-C 0|1

Disable error checking/correction:

0

No CRC check, no ECC correction (RAM).

1

CRC check, no ECC correction (SRAM or NOR Flash).

Copyright © 2014, QNX Software Systems Limited

Default: use CRCs for error checking and ECCs for error correction.

-c *nclusters*

Set the cache size. The cache holds recently read clusters in RAM, reducing the need to access the device if the same cluster is read again. It's also used to combine small writes into larger writes consisting of multicluster transactions. This reduces file fragmentation across the device and improves filesystem startup time. Since most devices are very fast, a small cache is usually acceptable. Larger caches may be desirable if many files are being written with small writes concurrently. The default value is 64 clusters, where cluster size is usually 1 KB or 2 KB, depending on the device.

-D

Specify driver options; see "*Driver options* (p. 804)," below.

-e

Erase the device and create an empty filesystem that is ready to use. For NAND flash, factory-marked bad blocks are not erased. Blocks that become bad during normal use (worn-out blocks) are also skipped during the erasing.

-F *num*

Defragment if the average extent is less than *num* clusters. The value of *num* must be in the range from 0 through 16. The default is 6. The `fs-etfs-ram` utility doesn't defragment if *num* is 0.

-f *numfiles*

Set the maximum number of files. The default value is 4096, with a maximum of 32,767.

Filenames that are more than 32 bytes long use two directory entries, reducing the number of files that you can actually have.

-I

Perform internal integrity checks of internal data structures while the filesystem is running. This slows down the filesystem. Its main purpose is for debugging new drivers and new versions of the filesystem.

-k

Allow the automatic correction of the `.counts` file in an image created using *`mketfs`* (p. 1219). If the value of the `num_blocks` attribute wasn't specified correctly, the size of the `.counts` file on the image, which is used

Copyright © 2014, QNX Software Systems Limited

for wear-leveling the part, won't be correctly calculated. The size of this file is also not updated at runtime at any point, and as such, it will leave many blocks (potentially all blocks if `num_blocks` isn't used at all) with no wear-leveling enabled. The -k option automatically updates the size of the `.counts` file to cover all the blocks on the part.

-L

Don't perform `.lost+found` recovery at startup.

-m ***mountpoint***

Set the directory for `fs-etfs-ram` to use as its mountpoint. On an embedded system where ETFS is the major filesystem, this is set as -m / for taking over the root. If you don't specify this option, the ETFS isn't mounted.

-o ***numattr***

Set the number of attributes to cache, which speeds up opens slightly. The default is 8.

-R

Reserve a percentage of the flash to avoid issues that arise when a flash device becomes completely full. Default is 5% (of the device size).

-r ***kbytes***

Set the size of the raw partition `/dev/etfs1`, in kilobytes. This partition, if present, is typically used to hold a boot image made using the *mkifs* (p. 1241) utility. The default size is 0.

-S

Implement transaction checksum using a fast and simple sum calculation rather than the default polynomial CRC algorithm. This may be faster but less robust.

-s ***num***

Set the number of flash blocks to use as spares. One spare block is required to perform a reclaim. During normal operation, flash devices wear, which causes flash blocks to fail. More than one spare block provides extra redundancy. The default is 4.

-t ***sec***

Set a timer for background operations. The default is 5 seconds.

-V

Copyright © 2014, QNX Software Systems Limited

Request read verifications on all writes.

-v[v...]

Be verbose. Each -v increases the verbosity.

-W *erasediff*

Set the wear-leveling span. Allow flash blocks to have erase counts that differ by more than *erasediff* before attempting to either :

- force them into service if they are below *erasediff*

  Or:

- give them a rest if they are above *erasediff*.

The default value is 50.

-x *nextents*

Cache this number of file extent offsets. This option reduces the processing needed to read through file extents on the device. Default is 8.

Driver options

use

Get a list of driver-specific options. This option causes the filesystem to print a usage message and then terminate without accessing the device.

size=*nn*M

Set the size of the RAM disk to *nn* megabytes. The default is 16 MB.

Description:

The embedded transaction filesystem (ETFS) implements a high-reliability filesystem for use with embedded solid-state memory devices with particular attention to NAND flash memory. The filesystem supports a fully hierarchical directory structure with POSIX semantics as shown in the table below:

POSIX capability	Supported by ETFS
Access date	Yes (if enabled with -a command-line option)
Modification date	Yes
Status change date	Yes
Max filename length	91 characters

Copyright © 2014, QNX Software Systems Limited

POSIX capability	Supported by ETFS
User permission	Yes
Group permissions	Yes
Other permissions	Yes
Directories	Yes
Hard links	No
Symbolic links	Yes

When started, ETFS creates two devices as follows:

/dev/etfs1

Raw partition for boot image

/dev/etfs2

Filesystem partition for `etfs` files.

The raw partition is used for boot images and is always at the start of the device. It may be zero bytes long if no boot image is needed. The filesystem partition is mounted in the pathname space as specified by the -m option.

If you don't specify the -m option, the filesystem isn't mounted. You can use the *mount* (p. 1312) command to mount it later:

```
mount -tetfs /dev/etfs2 my_mountpoint
```

ETFS is a filesystem composed entirely of *transactions*. Every write operation, whether of user data or filesystem metadata, consists of a transaction. A transaction either succeeds or is treated as if it never occurred.

For more information, see “Embedded transaction filesystem (ETFS)” in the Filesystems chapter of the *System Architecture* guide.

Examples:

Start ETFS to implement a temporary filesystem in RAM mounted at `/tmp`. Since it's not persistent across a reboot, and since RAM is reliable, you should disable all data error detection and correction (-C 0). The -e option initializes an empty filesystem ready to go upon startup. Since the filesystem is in high-speed RAM, you should specify the smallest cache possible with the -c 0 option.

```
fs-etfs-ram -C 0 -e -c 0 -m /tmp
```

Copyright © 2014, QNX Software Systems Limited

Caveats:

Although ETFS supports most POSIX semantics, some functionality isn't implemented in order to keep the driver simple and efficient. The unsupported POSIX semantics include:

- Hard links, and related links. For example the `.` and `..` directories aren't returned in a *readdir()*. Symlinks are fully supported.
- Access times aren't updated on the media unless the -a option is specified, to reduce flash writes.
- The parent directory's modification time isn't updated when the directory content changes (files are created or deleted).

Copyright © 2014, QNX Software Systems Limited

fs-ext2.so

Linux Ext2 filesystem

Syntax:

driver ... ext2 *ext2_options* ... &

where *driver* is one of the `devb-*` drivers.

Runs on:

QNX Neutrino

Options:

unfixbadver

Turns off work-arounds for corrupt volumes created by the buggy 1.19 version of `mke2fs`.

Description:

The Ext2 filesystem (`fs-ext2.so`) provides transparent access to Linux disk partitions. This implementation supports the standard set of features found in Ext2 versions 0 and 1.

Sparse file support is included in order to be compatible with existing Linux partitions. Other filesystems can only be "stacked" read-only on top of sparse files. There are no such restrictions on normal files.

If an Ext2 filesystem isn't unmounted properly, a filesystem checker is usually responsible for cleaning up the next time the filesystem is mounted. Although the `fs-ext2.so` module is equipped to perform a quick test, it automatically mounts the filesystem as read-only if it detects any significant problems (which should be fixed using a filesystem checker).

The following features are *not* currently supported:

- file fragments (sub-block allocation)
- large files (> 2 GB)
- filetype extension
- compression
- b-tree directories

Copyright © 2014, QNX Software Systems Limited

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Caveats:

Although Ext2 is the main filesystem for Linux systems, we don't recommend using `fs-ext2.so` as a replacement for the QNX 4 filesystem (`fs-qnx4.so`). Currently, we don't support booting from Ext2 partitions. Also, the Ext2 filesystem relies heavily on its filesystem checker to maintain integrity; this and other support utilities (e.g., `mke2fs`) aren't currently available for QNX Neutrino.

Copyright © 2014, QNX Software Systems Limited

fs-mac.so

Shared object that supports Apple Macintosh HFS and HFS Plus (QNX Neutrino)

Syntax:

driver … mac *mac_options*… &

Runs on:

QNX Neutrino

Options:

There are currently no *mac_options* defined.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *io-blk.so* (p. 993).

Description:

The fs-mac.so shared object provides read-only support for Apple HFS (Hierarchical File System) and HFS Plus. It's automatically loaded by the devb-* drivers when mounting an Apple filesystem.

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of EILSEQ).

The slash (/) character (which is valid in file names on the Mac but not in POSIX) is swapped with the colon (:) (which is the Mac path separator). For example, an HFS file called 29/1/2009 will get shown to ls as 29:1:2009, and when opened with that name will internally match back to the 29/1/2009 on-disk name.

The fs-mac.so shared object doesn't support access to the resource fork or hard links.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)

Copyright © 2014, QNX Software Systems Limited

Partition type	Filesystem	Shared object	Initialize with:	Check with:
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

fs-nfs2

NFS 2 client filesystem (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
fs-nfs2 [-b num] [-B size] [-D] [-e] [-h]
        [-i nodes] [-r] [-S] [-t] [-u] [-v[v]...]
        [-Z n] [server:export] [mountpoint]
        [[-erStu] [-Z n] server:export mountpoint ...]
```

Runs on:

QNX Neutrino

Options:

server

The name of the NFS server.

export

The directory to be exported from the server.

mountpoint

The name under which the exported directory is to be mounted.

The following options apply to all mountpoints:

-b *num*

Use *num* buffers (default: 200).

-B *size*

Set the buffer size to *size* bytes. The default is set by the first server, and is usually 8K.

-D

Run in the foreground.

-h

Copyright © 2014, QNX Software Systems Limited

Display usage information.

-i *nodes*

Set the number of inodes to *nodes*.

-v[v]...

Verbose output; add more v characters for more verbosity. In order to capture the log messages, you need to have *syslogd* (p. 1885) running.

The following options apply only to the next mountpoint specified on the command line:

-e

Set the NO EXEC flag for the mounted filesystem.

-r

Set the READ ONLY flag for the mounted filesystem.

-S

Don't cache symlinks.

-t

Use TCP instead of UDP. If this fails, `fs-nfs2` uses UDP.

-u

Use UDP (which is the default). If this fails, `fs-nfs2` fails.

-Z *n*

The value of *n* indicates how to attach to the path:

- B or b — attach before other managers.
- A or a — attach after other managers.
- O or o — make the attachment opaque; don't resolve to mountpoints with shorter pathname matches. The pathname resolver tries to find the longest match against all pathnames attached.

The default is none of these. For more information, see "Ordering mountpoints" in the Process Manager chapter of the *System Architecture* guide.

Description:

The `fs-nfs2` filesystem manager is an NFS 2 client operating over TCP/IP. To use it, you must have an NFS server.

Copyright © 2014, QNX Software Systems Limited

This filesystem manager requires a TCP/IP transport layer, such as the one provided by *io-pkt** (p. 1007). It also needs `socket.so` and `libc.so`.

By default, this utility does not set any upper limit for number of *inodes*.

You can also create mountpoints with the *mount* (p. 1312) command by specifying `nfs` for the type. You must start `fs-nfs2` before creating mountpoints in this manner. If you start `fs-nfs2` without any arguments, it runs in the background so you can use `mount`.

Examples:

Mount the `qnx_bin` export as `/bin` from an NFS server named `server_node`:

```
fs-nfs2 server_node:/qnx_bin /bin &
```

Mount `/nfs1` using TCP, and `/nfs2` using UDP:

```
fs-nfs2 -t host1:/ /nfs1 host2:/ /nfs2
```

Mount both using TCP:

```
fs-nfs2 -t host1:/ /nfs1 -t host2:/ /nfs2
```

Caveats:

If possible, you should use *fs-nfs3* (p. 814) instead of `fs-nfs2`.

Copyright © 2014, QNX Software Systems Limited

fs-nfs3

NFS 3 client filesystem (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
fs-nfs3 [-b num] [-B size] [-D] [-e] [-h]
        [-i nodes] [-r] [-s] [-S] [-t]
        [-T threads] [-u] [-v[v]...] [-w delay=sec]
        [-w number=num] [-w size=num] [-w sync=hard]
        [-Z n] [server:export] [mountpoint]
        [[-erStu] [-Z n] server:export mountpoint ...]
```

Runs on:

QNX Neutrino

Options:

server

The name of the NFS server.

export

The directory to be exported from the server.

mountpoint

The name under which the exported directory is to be mounted.

The following options apply to all mountpoints:

-b *num*

Use *num* buffers (default: 200).

-B *size*

Set the buffer size to *size* bytes. The default is set by the first server, and is usually 8K.

-D

Run in the foreground.

-h

Copyright © 2014, QNX Software Systems Limited

Display usage information.

-i *nodes*

Set the number of inodes to *nodes*.

-v[v]...

Verbose output; add more v characters for more verbosity. In order to capture the log messages, you need to have *syslogd* (p. 1885) running.

The following options apply only to the next mountpoint specified on the command line:

-e

Set the NO EXEC flag for the mounted filesystem.

-r

Set the READ ONLY flag for the mounted filesystem.

-S

Don't cache symlinks.

-s

Use a soft mount.

-t

Use TCP instead of UDP. If this fails, `fs-nfs3` uses UDP.

-T *num*

Set the number of threads. The default is 5.

-u

Use UDP (which is the default). If this fails, `fs-nfs3` fails.

-w delay=sec

Indicate the time, in seconds, after which the data will be flushed to the server. The default is 2 seconds.

-w number=num

Number of buffers (default is 10.) Each buffer manages content for one file. The default value indicates 10 files can be buffered simultaneously.

-w size=num

Size of the buffer, in units of 1K (default is 8.)

Copyright © 2014, QNX Software Systems Limited

-w sync=hard

Turn off write caching.

-Z *n*

The value of *n* indicates how to attach to the path:

- B or b — attach before other managers.
- A or a — attach after other managers.
- O or o — make the attachment opaque; don't resolve to mountpoints with shorter pathname matches. The pathname resolver tries to find the longest match against all pathnames attached.

The default is none of these. For more information, see "Ordering mountpoints" in the Process Manager chapter of the *System Architecture* guide.

Description:

The `fs-nfs3` filesystem manager is an NFS 3 client operating over TCP/IP. To use it, you must have an NFS server.

When you use `fs-nfs3` with write caching (the default), you benefit from enhanced filesystem performance. However, there can be interoperability issues if more than one NFS client accesses the same file on the NFS server. If `fs-nfs3`'s cached data hasn't been written to the NFS server, another NFS client attempting to read the same file won't see the changes to the file until they're written to the server at a later point. If you want `fs-nfs3` to write file modifications immediately to the NFS server, use the -w sync=hard option to turn off write caching.

This filesystem manager requires a TCP/IP transport layer, such as the one provided by *`io-pkt*`* (p. 1007). It also needs `socket.so` and `libc.so`.

By default, this utility does not set any upper limit for number of *inodes*.

You can also create mountpoints with the *`mount`* (p. 1312) command by specifying `nfs` for the type and -o ver3 as an option. You must start `fs-nfs3` before creating mountpoints in this manner. If you start `fs-nfs3` without any arguments, it runs in the background so you can use `mount`. The options that you can use with `mount` include the following:

tcp

Use TCP instead of UDP. If this fails, `mount` uses UDP.

udp

Use UDP (which is the default). If this fails, `mount` fails.

Copyright © 2014, QNX Software Systems Limited

nocachesymlink

Don't cache symlinks.

ver3

Use `fs-nfs3` instead of `fs-nfs2`.

soft

Use a soft mount (i.e. break the connection if unable to reach the server).

Examples:

Mount the `qnx_bin` export as `/bin` from an NFS server named `server_node`:

```
fs-nfs3 server_node:/qnx_bin /bin &
```

Mount `/nfs1` using TCP, and `/nfs3` using UDP:

```
fs-nfs3 -t host1:/ /nfs1 host2:/ /nfs3
```

Mount both using TCP:

```
fs-nfs3 -t host1:/ /nfs1 -t host2:/ /nfs3
```

Mount an NFS filesystem (`fs-nfs3` must be running first):

```
mount -t nfs -o ver3 server_node:/qnx_bin /bin
```

Mount an NFS filesystem, using TCP (`fs-nfs3` must be running first):

```
mount -t nfs -o tcp,ver3 server:/tmp /mnt
```

Caveats:

If possible, you should use `fs-nfs3` instead of `fs-nfs2`.

Copyright © 2014, QNX Software Systems Limited

fs-nt.so

Shared object that supports the Windows NT filesystem (QNX Neutrino)

Syntax:

driver … nt *nt_options*… &

Runs on:

QNX Neutrino

Options:

compress=*number*

The number of buffers to use for the dynamic decompression of compressed files (e.g. if in a directory you set the "Compress contents to save disk space" folder attribute). The default is 2; a value of 0 disables support for compressed files.

dots=[offlon]

Fabricate `.` and `..` directory entries. The default is off.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *io-blk.so* (p. 993).

Description:

The `fs-nt.so` shared object provides read-only support for the Microsoft Windows NT filesystem. It's automatically loaded by the `devb-*` drivers when mounting an NT filesystem.

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).

The `fs-nt.so` shared object doesn't support encrypted files, security IDs or ACLs, and alternate data streams.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Copyright © 2014, QNX Software Systems Limited

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

fs-qnx4.so

QNX 4 compatible filesystem support

Syntax:

```
driver ... qnx4 qnx4_options... &
```

Runs on:

QNX Neutrino

Options:

Where *driver* is any of the `devb-*` drivers, and *qnx4_options* is one or more of the following, separated by commas:

bitmap=*when*

When to pre-read the `.bitmap` file. Scanning the bitmap is required to return the count of free blocks. It also allows for improved *write()* performance by creating an in-memory summary of where free blocks might be located within the filesystem.

The value must be one of the following:

- `always` — calculate/store `.bitmap` details for all media.
- `lazy` — don't read the bitmap for non-removable media at mount time, but when it's first needed (e.g. by a *statvfs()* call or *df* (p. 476)). This can help speed up system boot times (since on a large disk, the `.bitmap` files may be several megabytes in length, and if read right away, can interfere with the starting of other processes on an embedded system).
- `nonrmv` (the default) — act as `always` for nonremovable media, and `lazy` for removable media

grown

Allow persistent over-grown files; don't truncate them when they're closed. Certain file-write access patterns (e.g `O_APPEND`) are detected, and the file isn't shrunk back at the last close. This is useful for log files that you keep appending to, and so on.

noembed

Never embed inode details; always place in fixed-size `.inodes`.

overalloc

Copyright © 2014, QNX Software Systems Limited

Enable a more aggressive file-extent over-allocation heuristic.

unbusy

Attempt to repair any file marked as "busy" on the filesystem (i.e. a file that was being grown or shrunk when the system was improperly shutdown). The default action is to return EBADFSYS to any attempt to open such a file; this option will instead truncate the file to its last-known valid size, unset the "busy" indicator, and allow access. This truncation may result in lost data and unused blocks marked as used in the bitmap, so run `chkfsys` later to ensure full filesystem consistency.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *`io-blk.so`* (p. 993).

Description:

The `fs-qnx4.so` shared object provides support for QNX 4 filesystems. It's automatically loaded by the `devb-*` drivers when mounting a QNX 4 filesystem.

The maximum numeric group or user ID on a QNX 4 filesystem is 65534.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
7	Windows NT[a]	*`fs-nt.so`* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*`fs-dos.so`* (p. 795)	*`mkdosfs`* (p. 1205)	*`chkdosfs`* (p. 111)
77, 78, or 79	QNX 4	*`fs-qnx4.so`* (p. 820)	*`dinit`* (p. 626)	*`chkfsys`* (p. 114)
131	Linux (Ext2)	*`fs-ext2.so`* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*`fs-mac.so`* (p. 809)	N/A	N/A

Copyright © 2014, QNX Software Systems Limited

Partition type	Filesystem	Shared object	Initialize with:	Check with:
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Files:

`.longfilenames`

To enable support for long filenames (more than 48 characters) on an existing QNX 4 compatible filesystem, login as `root` and create an empty, read-only file named `.longfilenames` in the root directory of that filesystem.

To enable support for long filenames on a new QNX 4 filesystem, use the -N option to *dinit* (p. 626).

Copyright © 2014, QNX Software Systems Limited

fs-qnx6.so

Shared object that supports the Power-Safe filesystem (QNX Neutrino)

Syntax:

driver … qnx6 *qnx6_options*… &

Runs on:

QNX Neutrino

Options:

The *driver* is any of the devb-* drivers, and *qnx6_options* is one or more of the following, separated by commas:

alignio

Attempt to align all reads and writes in sizes and offsets of the filesystem block size.

crypto=enable|disable

Enable or disable encryption support (disabled by default). In order to use encryption, you must have formatted the filesystem with the -E option for *mkqnx6fs* (p. 1274). Use *fsencrypt* (p. 834) to manage the encryption.

hold=allow|root|deny

Control which users (if any) can suspend the taking of snapshots (via a flag in the DCMD_FSYS_FILEFLAGS *devctl()* command). The default is root.

overalloc

Enable a block overallocation heuristic for small file writes.

snapshot=*freq*

Set the frequency of automatic snapshots; the default is 10 seconds. A filesystem snapshot is explicitly made when you call *sync()* or *fsync()*, or from this periodic timer.

sync=*mode*

Specify the required disk synchronization capability. The *mode* mode must be one of the following:

Copyright © 2014, QNX Software Systems Limited

- `mandatory` (the default) — the drive must support synchronization to allow a filesystem to be mounted read/write. If it doesn't, the *`mount`* (p. 1312) fails and returns EROFS. A read-only mount (`mount -r`) can always be performed on any device.
- `optional` — attempt synchronization, but ignore any error if the drive doesn't support such an operation. The driver might be incorrectly advertising the capabilities, or the physical media might not require explicit synchronization (write-through).
- `none` — never issue a synchronization command to the disk, and don't drain dirty blocks from the filesystem cache (until an explicit *`umount`* (p. 2009)). This mode is suitable only for use with a UPS.

If the drive doesn't support synchronizing, `fs-qnx6.so` can't guarantee that the filesystem is power-safe. You can use the sync option to override this requirement at your own risk. Before using this filesystem on devices — such as USB/Flash devices — other than traditional rotating hard disk drive media, check to make sure that your device meets the filesystem's requirements. For more information, see "*Required properties of the device* (p. 825)," below.

trim=disablelenableldiscard

Disable or enable support for TRIM, or use discard instead.

A managed NAND block device can't overwrite in-place and has no idea of whether content in a block is even valid or meaningful to a mounted filesystem. So the management layers have no choice but to preserve all written content, which can be a lot of wear-levelling overhead if in fact those blocks belonged to say a deleted file, or if the partition was freshly formatted.

The TRIM command is thus a hint to the managed NAND device from the filesytem that certain sectors are no longer live and can be discarded (i.e., the content doesn't have to be preserved or copied by wear-levelling, and/or logical blocks can be erased rather than be reclaimed from elsewhere).

Using the discard option gives better performance than enabling trim. When the filesystem tells the driver to discard a set of blocks, the driver simply marks them as discarded and returns, queuing them up for garbage collection later. If the filesystem requests the driver to trim a set of blocks, they're cleaned immediately, which may result in heavy disk I/O, depending on the current state of the system. In the end they do the same thing, just with different timing.

Copyright © 2014, QNX Software Systems Limited

Description:

The `fs-qnx6.so` shared object provides support for Power-Safe (copy-on-write/snapshot) filesystems. It's automatically loaded by the `devb-*` drivers when mounting a Power-Safe filesystem.

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).

Required properties of the device

The Power-Safe filesystem was designed for and is intended for traditional rotating hard disk drive media. It operates by moving the on-disk filesystem state from one stable view to another stable view using copy-on-write (COW) to relocate modified blocks. To finalize this transition, all dirty blocks involved in the new view must be committed to persistent storage, and then a new filesystem superblock/root referencing the relocated blocks is committed.

This provides power-safe robustness, because at any point in time either the old version is completely accessible or the new version is completely accessible (with no live data being overwritten in between). Thus to mount as read-write on a given device, that device must have the following properties:

- *one* of the following:
  - The device may buffer write data for performance reasons, and the return from a WRITE may not necessarily indicate the data is committed to permanent storage. But such a device must implement a FLUSH/SYNC command that forces any cached or buffered write data to persistent storage, and doesn't return until it's guaranteed that all data is stable across a power-loss.

    or:
  - The device doesn't buffer write data, and operates in a strict write-through manner, where return from a WRITE is a guarantee that the data was immediately committed to persistent storage. Such a device doesn't require an additional FLUSH/SYNC command.
- and *both* of the following:
  - The action of writing to one data region (an advertised device sector) can in no way damage the contents of any other region, even under conditions such as power-loss, vibration, temperature, etc.

    and:
  - Data that has previously been reported as committed to persistent storage remains stable until explicitly overwritten. The device may implement facilities such as bad-block remapping or wear-leveling to support this requirement,

Copyright © 2014, QNX Software Systems Limited

provided that such activity never causes loss of persistent data, even under conditions such as power-loss, etc.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

fs-rcfs.so

Shared object that supports the read-only compressed filesystem (QNX Neutrino)

Syntax:

driver … rcfs *rcfs_options*… &

Runs on:

QNX Neutrino

Options:

cache-limit=*N*

Set to the size of KiB allowed to cache the inode and file name table.

cbuf-ceiling=*N*

The maximum number of compression buffers that may be allocated at run time.

cbuf-disable

Minimize the number of compression buffers allocated.

cbuf-floor=*N*

The minimum number of compression buffers that are allocated at startup and maintained at runtime.

ext-ninodes=*N*

The default number of inodes to allocate for an extended (write-capable) volume.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *io-blk.so* (p. 993).

Description:

The fs-rcfs.so shared object supports the read-only compressed filesystem. It's automatically loaded by the devb-* drivers when mounting an RCFS.

It's a read-only filesystem that compresses the files in blocks, and still supports random access. The filesystem can be mounted from a disk partition, or from a file. The maximum file size supported is 2 GB, and the maximum filesystem size is 4 GB.

Copyright © 2014, QNX Software Systems Limited

The RCFS has minimal write support; to enable write support, use the *mount -uw* (p. 1312) command. Random writes aren't supported, but you can:

- delete files from the filesystem and replace them with new files
- add new files one at a time
- append to the most recently written file

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

fs-udf.so

Universal Disk Format and ISO 9660 filesystem support

Syntax:

```
driver ... udf udf_options ... &
```

Runs on:

QNX Neutrino

Options:

Where *driver* is one of the `devb-*` drivers, and *udf_options* is one or more of the following, separated by commas:

case=asis|lower|upper

Control the case used to display ISO 9660 filenames (to a *readdir()* request; all pathname matching is always performed case-insensitively on such names):

- `asis` — don't convert the filename in any way; if the CD was mastered with strict ISO 9660 compliance, the name will be in uppercase, but more lenient utilities could produce mixed-case filenames.
- `lower` — convert to lowercase (the default).
- `upper` — convert to uppercase.

RRIP, Joliet, and ISO 9660:1999 store case-preserving names and ignore this option.

If you specify both the case and charset options for an ISO 9660 format, `fs-udf.so` ignores the case option.

charset=[*pvd_charset*][:*svd_charset*]

Use a non-standard character set mapping for ISO 9660:1988 Primary Volumes and ISO 96660:1999 Supplementary Volumes. By default, these are ISO 646 (ASCII). However a number of CD-burning tools incorrectly use extended characters in filenames. An appropriate charset= specification can allow these filenames to be displayed according to a known alternate character set; without such a hint, `fs-udf.so` replaces with an underscore (_) any illegal characters in filenames.

Copyright © 2014, QNX Software Systems Limited

Enhanced SVD format allows the active character set to be specified via ISO 2022 escape sequences, but again many CD-burning utilities fail to do this correctly, thus separate overrides are provided. The supported character sets include:

- `IBM850`
- `IBM852`
- `IBM866`
- `US-ASCII`
- `ISO-8859-1`
- `ISO-8859-2`
- `windows-1250`
- `windows-1251`
- `windows-1252`

The case in these strings doesn't matter.

fileset=*num*

The File Set number to mount; the default is 0.

format=*list*

Set both the list of disk formats to support, as well as the order in which they should be probed (for media with multiple formats, such as UDF-Bridge/DVD-Video or ISO/Joliet). Separate the formats with colons (`:`) You can use this option to:

- set a specific order in which to probe (e.g. format=joliet:rrip:udf)
- remove a format (e.g. format=-rrip)
- add a format, making it the first preference (e.g. format=+udf)

The valid formats are:

- `udf` — OSTA/UDF, all v1.x and 2.x variants as supported.
- `rrip` — Rock Ridge extensions to ISO 9660; adds permissions and long names.
- `joliet` — Joliet extensions to ISO 9660; adds Unicode long names.
- `iso9660e` — the 1999 version of the ISO 9660 spec; adds mixed-case filenames.
- `iso9660` — the base 1988 version of the ISO 9660 spec.
- `audio` — create a dummy mountpoint for an audio-only/CDDA disk.

The first matching, valid format in order from the specified list is mounted.

Copyright © 2014, QNX Software Systems Limited

Since the `audio` format matches any disk with audio tracks, you should usually make it the last in the list. In addition, since many formats are extensions to a base ISO 9660 format, which is also present on the media, you should specify `iso9660` itself after those formats.

The default is format=udf:rrip:joliet:iso9660e:iso9660:audio. For backward compatibility, set the format to format=udf to disable the CD/ISO formats.

gid=*group*

The group ID to use for files with no specified group. The default is 0.

hidden=*hidden_mode*

Specify what to do with "hidden" files. The *hidden_mode* can be one of:

- `ignore` — ignore the hidden files; they don't appear in the filesystem.
- `show` — (the default) show hidden files in the filesystem as normal files.
- `dot` — show hidden files in the filesystem with a dot (`.`) prefixed to their names.

info=*path*

The name of the filesystem metadata directory. The first character can be + or -, and this controls whether empty entries (metadata descriptors not given a value) are shown in the directory or not, respectively. For example, if a CD doesn't have a abstract or bibliography, those pseudo-files can be hidden, or left with a empty string in them.

The default is `-.info.`.

perms=[*file_permissions*][:*directory_permissions*]

The permissions to use for ISO 9660 files, directories, or both. The argument to this option consists of the permissions for files, followed by a colon (`:`), and then the permissions for directories. Either set of permissions is optional.

You can specify the permissions either as a simple numeric value, or in *chmod* (p. 124)-style format. For example, to make files executable, specify perms=+x, which is the equivalent to the exe option to *fs-cd.so* (p. 787). Like the uid and gid options, this option is used only when the filesystem itself doesn't have explicit permissions (`udf` and `rrip` do; all others don't).

The default is a=r:a=rx.

raw=*num*[:*chunk*]

Copyright © 2014, QNX Software Systems Limited

Set the number of raw CDDA/CDXA 2352-byte buffers and optionally the number of blocks to read with one raw I/O operation. The default is `0:16` (raw read presentation as described below is disabled).

If you specify only *num* (e.g. raw=2 vs raw=2:8), then the chunk size remains 16. The size of your raw buffer is the product of these two numbers and 2352. Reading more sectors per operation improves the overall read performance.

When enabled, this option supports the transparent reading of Mode 2 Form 2 VCD and audio files:

- For Mode 2 Form 2 VCD files, a 44-byte RIFF header is constructed and prepended to the file data, and then data from the files' raw 2352-byte sectors are supplied.
- Any audio tracks also get a filename in `.info./CD_Track`*NN*`.wav`, which when read yields a similar RIFF header (but WAVE instead of CDXA format), followed by the raw ripped audio.

If this option is disabled, such files are unreadable to POSIX *read()* and fail with `EINVAL`.

You can use the *`chattr`* (p. 107) command to identify Mode 2 Form 2 or CDDA files.

uid=*user*

The user ID to use for files with no specified owner. The default is 0 (`root`).

verify=*level*

How much of the ISO PVD or UDF tag content (tag version, tag location, header checksum, and/or data CRC) to verify; one of the following:

- `none`
- `tag` — all except the data CRC
- `all`
- `?none`
- `?tag`
- `?all` (the default)

Some ISO 9660:1999 SVD-mastering utilities and UDF-authoring utilities write incorrect tags, so it might be necessary to relax the verification if problems are observed. If you specify an option with a leading question mark (?), `fs-udf.so` consults an internal blacklist of known bad utilities and automatically skips all checks in the list; otherwise it operates at the level given after the question mark.

Copyright © 2014, QNX Software Systems Limited

volume=*num*

The Primary Volume number to mount; the default is 0.

In addition, you can specify any of the *filesystem options* (p. 1000) described for *io-blk.so* (p. 993).

Description:

The `fs-udf.so` shared object provides support for UDF (OSTA-UDF/ECMA-167) and ISO 9660 (base 1988 spec, 1999 spec, Joliet extensions, Rock Ridge extensions) filesystems. It's automatically loaded by the `devb-*` drivers when mounting a UDF filesystem.

This filesystem uses UTF-8 encoding for presentation of its filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`).

Copyright © 2014, QNX Software Systems Limited

fsencrypt

Filesystem encryption manager

Syntax:

```
fsencrypt -c cmd [-d domain] [-f] [-K .|:|+|#|@key [-ooffset]]
          [-k .|:|+|#|@key [-ooffset]] [-l log_path]
          [-n value] -p path [-r] [-t type] [-v[v...]]
```

Runs on:

QNX Neutrino

Options:

-c *cmd*

The command to run (see below).

-d *domain*

The domain number to be used (1–100).

-f

If *path* is a directory, make the move or remove action on the files as well.

-K *key*

Specify a secondary key, in the same form as for -k.

-k *key*

Specify key data in one of the following forms:

- *.salt.str* — a 64-bit salt value expressed as a string of bytes in hexadecimal digits that may be postfixed to a plain-text string
- :*setup* — a command string used with the `setup` command. The string format is:

  domain:*type*:*locked*:*provider*:*path*

- +*str* — a user-supplied plain-text string, to be hashed into a 512-bit key
- #*str* — a base-64 representation of a key, which must be 512 bits long
- @*file* — the name of a file that contains binary key data, which must be 512 bits long

Copyright © 2014, QNX Software Systems Limited

-l ***log_path***

The path of the log file to use (*stdout* is the default).

-n ***value***

Specify a secondary value that some commands require.

-o ***offset***

("oh") An offset into a key file specified with the -K@ or -k@ option.

-p ***path***

The mountpoint of a Power-Safe (*fs-qnx6.so* (p. 823)) filesystem, or a file in the filesystem, depending on the command.

-r

If *path* is a directory, take action on the entire tree.

-t ***type***

Used in the creation of a domain to set the encryption mechanism. The supported types include:

- 0 — no encryption
- 1 — AES-256, in XTS mode. The two keys are randomly generated.
- 2 — AES-256, in CBC mode

-v[v...]

Be verbose; each v increases the level of verbosity. If you don't turn on verbosity, some commands indicate success or failure only by `fsencrypt`'s exit status.

Description:

The `fsencrypt` utility manages the encryption of a Power-Safe (*fs-qnx6.so* (p. 823)) filesystem. In order to use `fsencrypt`, you must have formatted the filesystem with the -E option for *mkqnx6fs* (p. 1274), and then specified crypto=enable for `fs-qnx6.so`.

The commands that you can specify with the -c option are given below, along with the other options that you must specify for each command:

change-key

Change a domain key:

```
fsencrypt -p path -c change-key -d domain -k old_key -K new_key
```

check

Check for support of encryption on a given filesystem:

```
fsencrypt -p path -c check
```

check-key

Verify that the key given is is valid against a domain:

```
fsencrypt -p path -c check-key -d domain -k key
```

create

Create a domain:

```
fsencrypt -p path -c create -d domain -k key -t type
```

The new domain is unlocked.

destroy

Destroy a domain. The domain must be unlocked, and you must be in the group that owns the mountpoint:

```
fsencrypt -p path -c destroy -d domain
```

If you destroy a domain, you won't be able to access any of its contents because they'll be encrypted and the domain's encryption key will have been destroyed. The contents remain in the filesystem until you delete them.

enable

Enable encryption support on a volume that wasn't set up for it at formatting time:

```
fsencrypt -p path -c enable
```

get

Copyright © 2014, QNX Software Systems Limited

Determine the domain that the given *path* belongs to:

```
fsencrypt -p path -c get
```

lock

Lock a domain within the given filesystem:

```
fsencrypt -p path -c lock -d domain
```

migrate-delay

Change the migration delay between work units. Use the -n option to indicate a period in milliseconds.

```
fsencrypt -p path -c migrate-delay -n milliseconds
```

migrate-path

Parse a path, assigning the given domain to directories and tagging files to the given domain:

```
fsencrypt -p path -c migrate-path -d domain
```

migrate-state

Determine the amount of remaining migration work:

```
fsencrypt -p path -c migrate-state
```

migrate-start

Start the background encryption of tagged files:

```
fsencrypt -p path -c migrate-start
```

migrate-status

Report the status of migration:

```
fsencrypt -p path -c migrate-status
```

migrate-stop

Copyright © 2014, QNX Software Systems Limited

Suspend the background encryption migration:

```
fsencrypt -p path -migrate-stop
```

migrate-tag

Tag a file for migration into the given domain (`tag` is a synonym):

```
fsencrypt -p file -c migrate-tag -d domain
```

migrate-units

Set the amount of work to complete beween delay periods. Use the -n option to indicate a number of blocks:

```
fsencrypt -p path -c migrate-units -n blocks
```

query

Query the status of a domain:

```
fsencrypt -p path -c query -d domain
```

query-all

Query the status of all the domains for a filesystem

```
fsencrypt -p path -c query-all
```

read-key

Read a file key into *file*.

```
fsencrypt -p path -c read-key -k @file
```

set

Assign the given *path* to a numbered domain. The domain must be unlocked.

```
fsencrypt -p path -c set -d domain
```

set-whole-disk

Copyright © 2014, QNX Software Systems Limited

Enable whole-disk encryption using *domain*. There must be only one domain:

```
fsencrypt -p path -c set-whole-disk -d domain
```

Plain-text files are hidden if you enable whole-disk encryption.

setup

Complete the domain setup based on the provided -k :*str*.

```
fsencrypt -p path -c setup -k :domain:type:locked:provider:path
```

tag

Tag a file for migration into the given domain (`migrate-tag` is a synonym):

```
fsencrypt -p file -c tag -d domain
```

unlock

Unlock a domain, given the correct key data:

```
fsencrypt -p path -c unlock -d domain -k key
```

write-key

Write a file key described by *file* to file at *path*:

```
fsencrypt -p path -c write-key -k @file
```

Examples:

Create domain 10 on the root volume using a plain-text password with a 64-bit salt value:

```
fsencrypt -vc create -d10 -t1 -p/ -k.1234567890abcdef.mypassword
```

Unlock the domain:

```
fsencrypt -vc unlock -d10 -p/ -k.1234567890abcdef.mypassword
```

Add a directory to this domain:

```
fsencrypt -vc set -d10 -p/secure_dir
```

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

Success.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

fsysinfo

Display filesystem statistics (QNX Neutrino)

Syntax:

```
fsysinfo [options]* device|filesystem
```

Runs on:

QNX Neutrino

Options:

-I *period*

Loop with the specified period (in milliseconds).

-L *period*

Loop with the specified period (in milliseconds), displaying the differences in the statistics between iterations.

-m

Don't add in metadata (filesystem mountpoint) disk I/O counts.

-v

Display the versions of `io-blk.so` and the filesystem.

-z

Zero (reset) the statistics after reading them.

-Z

Zero (reset) the statistics, without displaying them.

Description:

The `fsysinfo` utility is a front end to the `devctl(DCMD_FSYS_STATISTICS)` command (from `<sys/fs_stats.h>`), which queries a disk filesystem for its statistics. These statistics include disk sectors read and written, the efficiency of internal caches, the count of various system calls made from user code, and so on.

By default, the current statistics are displayed once. There's also a looping mode, in which the utility loops and redisplays the statistics at the specified interval (-l displays

Copyright © 2014, QNX Software Systems Limited

the raw values, and -L displays the difference from the previous values). Press **Ctrl**–**C** to exit.

You can reset the statistics to 0 by using -z or -Z (-z displays what the values were prior to being reset, but -Z produces no output and is useful in scripts).

If you specify the -m option, `fsysinfo` doesn't combine disk I/O statistics from the specified filesystem and its host device/partition; usually this is necessary for many filesystem implementations that internally target metadata updates against the host device and not to a particular higher-level file. (For example, if you specify -m, modifications to the `.bitmap` and `.inode` files for an `fs-qnx4` filesystem don't show as I/O on the mounted filesystem.)

The output includes the following:

Section	Statistic	Description
`FILESYS`		Mountpoint and type of filesystem
`MOUNT`	`mounted`	Time that the filesystem was mounted
	`elapsed`	Time since the statistics were reset
`DISK I/O`	`write`	Physical sector writes to the device
	`read`	Physical sector reads from the device
	`r/a`	Predictive readaheads from the device
	`direct`	`DCMD_FSYS_DIRECT_IO` sectors
	`bad`	The number of bad blocks
`CACHE`	`write`	Sectors delayed-write to buffer cache (blk delwri= commit=)
	`read`	Sectors reread from buffer cache
	`rate`	Ratio of (`CACHE read`) : (`DISK IO read` + `r/a`)
	`mru`	The size, in KB, of the Most Recently Used region

Copyright © 2014, QNX Software Systems Limited

Section	Statistic	Description
	mfu	The size, in KB, of the Most Frequently Used region
	ratio	Ratio of (CACHE mru) : (CACHE mfu)
SYSCALL	open	Number of *open()* calls
	stat	Number of *stat()* or *fstat()* calls
	namei	Number of conversions from a pathname to a file
	write	Number of *write()* calls
	read	Number of *read()* calls
	devctl	Number of *devctl()* calls
	create	Number of creations
	delete	Number of deletions
NAMES	exist	Filenames known to exist (avoid scan)
	enoent	Filenames known to not exist (avoid scan)
	misses	Filename unknown (have to scan directory)
	unsuit	Filename unsuitable for caching (too long or ambiguous)
	stale	Stale name entry (fsys unmounted or associated vnode recycled)
	rate	Ratio of (exist + enoent) : (misses + unsuit)
BMAP	hit	Known logical to physical block mapping
	miss	Unknown logical to physical block mapping
	rate	Ratio of (hit) : (miss)

Copyright © 2014, QNX Software Systems Limited

Section	Statistic	Description
VNODES	create	Open of unknown file (must load/build the vnode from the filesystem)
	hit	Open of a known or recently-used file
	rate	Ratio of (hit) : (create)
	lock	Number of locked vnodes
	recycl	Reuse of cache entry for new file (blk vnode=)
SLAB	map	The number of pages mapped in by the heap-allocation subsystem
	unmap	The number of pages released by the heap-allocation subsystem
	active	The difference of (map) – (unmap)

You can use this information to analyze the efficiency of the various `io-blk.so` caches, possibly leading to fine-tuning of the driver command-line options (blk cache= ncache= map= vnode=).

Examples:

```
# fsysinfo /tmp
FILESYS     / (qnx6)
MOUNT       mounted  Mon Sep 21 06:31:21 2009     elapsed 39 secs
DISK I/O    write          0   read           0   r/a             0
            direct         0   bad            0
CACHE       write        328   read       23696   rate          100%
            mru         6792k  mfu         2832k  ratio          29%
SYSCALL     open         1211  stat         1331  namei         2718
            write           3  read         2499  devctl         579
            create          3  delete          3
NAMES       exist        4960  enoent       1096  misses         652
            unsuit          8  stale           1  rate           90%
BMAP        hit          3977  miss          464  rate           89%
VNODES      create        345  hit          7698  rate           95%
            lock        17391  recycl          2
SLAB        map           114  unmap           2  active         112
```

Determine what/how many disk operations were required by an activity:

```
fsysinfo -Z /fs/hd/tmp
cd /fs/hd/tmp
...
fsysinfo /fs/hd/tmp
```


Copyright © 2014, QNX Software Systems Limited

Monitor disk access in realtime, updating the statistics every 500 ms:

```
fsysinfo -L500 /fs/hd
```

Copyright © 2014, QNX Software Systems Limited

/etc/fstab

File for predefined mountpoints

Name:

/etc/fstab

Description:

The /etc/fstab file contains descriptive information about filesystems. Programs read it, but don't write it; it's the duty of the system administrator to properly create and maintain this file. Each filesystem is described on a separate line; fields on each line are separated by tabs or spaces. Lines beginning with # are comments.

If you specify the -a option to the *mount* (p. 1312) command, the utility mounts the devices that are listed in /etc/fstab. The format of the /etc/fstab used with the mount utility is as follows:

```
specialdevice mountpoint type mountoptions
```

For example, the following entry in /etc/fstab:

```
/dev/hd0t77 /mnt/fs qnx4 rw
```

is equivalent to calling:

```
mount -t qnx4 /dev/hd0t77 /mnt/fs
```

The *mountoptions* field is a comma-separated list of values that must contain, at a minimum, one of ro or rw to indicate a read-only or a read-write mount.

By default, the mount is performed with the *type* as if the -t option had been specified (device and server doing the mount are the same) but to get the -T type behavior, you should specify allservers in the options.

The following sample /etc/fstab indicates the mapping of the different configurations:

```
#This is a sample file that shows the mapping of command line
#arguments to the fstab entries and how they would be invoked.
#The "implied" argument is not generally required, but some
#servers may differentiate between implied and specified entries.

# mount -b -vvv -t mytype /my/specialdev1 /my/mountpoint1
# mount -vvv /my/mountpoint1
/my/specialdev1 /my/mountpoint1 mytype rw

# mount -b -vvv -t mytype /my/specialdev2
# mount -vvv /my/specialdev2
/my/specialdev2 / mytype rw,implied

# mount -b -vvv -T mytype /my/specialdev3 /my/mountpoint3
# mount -vvv /my/mountpoint3
/my/specialdev3 /my/mountpoint3 mytype allservers,rw
```


Copyright © 2014, QNX Software Systems Limited

```
# mount -b -vvv -T mytype /my/specialdev4
# mount -vvv /my/specialdev4
/my/specialdev4 / mytype allservers,implied,rw
```

Copyright © 2014, QNX Software Systems Limited

ftp

Internet file transfer program (UNIX)

Syntax:

```
ftp [-46AadefginpRtVv] [-N netrc] [-o output] [-P port]
    [-q quittime] [-r retry] [-s srcaddr]
    [-T dir,max[,inc]] [[user@]host [port]]
    [[user@]host:[path][/]] [file:///path]
    [ftp://[user[:password]@]host[:port]/path[/][;type=X]]
    [http://[user[:password]@]host[:port]/path] [...]

ftp -u URL file [...]
```

Runs on:

QNX Neutrino

Options:

See below.

Description:

The `ftp` utility is the user interface to the standard Internet File Transfer Protocol. With this utility, you can transfer files to and from a remote network site. For more information, see the NetBSD documentation at *http://netbsd.gw.com/cgi-bin/man-cgi?ftp++NetBSD-5.0*.

Based on:

RFC 959, *RFC 1123*, *RFC 1738*, *RFC 2068*, *RFC 2428*, and *RFC 2732*

Copyright © 2014, QNX Software Systems Limited

/etc/ftpchroot

Access-control file for `ftpd`

Name:

`/etc/ftpchroot`

Description:

The `/etc/ftpchroot` file lists the users who should have their session's root directory changed (using *chroot()*). The root directory is changed accordingly:

- if set, the directory that's specified in the *`/etc/ftpd.conf`* (p. 853) `chroot` directive
- home directory of the user.

If this file doesn't exist, the root directory change isn't performed.

The syntax is similar to `/etc/ftpusers`, except that the *class* argument is ignored:

`userglob``[:`*`groupglob`*`][@`*`host`*`] [`*`directive`*`]`

Please refer to *`/etc/ftpusers`* (p. 858) for detailed descriptions for each element.

If there's a positive match, the session's root directory is changed. No further comparisons are attempted after the first successful match. This syntax is backward-compatable with the old syntax.

Copyright © 2014, QNX Software Systems Limited

ftpd

Internet file transfer protocol daemon (NetBSD)

Syntax:

```
ftpd [-46DdHlnQqrsUuWwX] [-a anondir] [-C user] [-c confdir]
     [-e emailaddr] [-h hostname] [-L xferlogfile]
     [-P dataport] [-V version]
```

Runs on:

QNX Neutrino

Options:

In addition to the options described in the NetBSD documentation, `ftpd` supports the following:

-n

Don't attempt to translate IP addresses into hostnames.

Description:

The `ftpd` daemon is an Internet File Transfer Protocol server. It uses the TCP protocol. For more information, see the NetBSD documentation at *http://netbsd.gw.com/cgi-bin/man-cgi?ftpd++NetBSD-4.0*.

Setting up a restricted `ftp` subtree

So that system security isn't breached, it's recommended that the `ftp` subtree be constructed with care; the following rules are recommended:

`~ftp`

Make the home directory owned by the superuser and unwritable by anyone.

`~ftp/bin`

Make this directory owned by the superuser and unwritable by anyone. Generally, conversion commands are installed here. The `ls` utility, which must be present to support the `LIST` command, should have mode `111`.

`~ftp/usr/lib`

A directory to contain shared libraries. This example uses `/usr/lib` — as it is usually part of `_CS_LIBPATH` (see `getconf _CS_LIBPATH`); however, this may vary on custom installations. If no binaries in `~ftp/bin` use shared

Copyright © 2014, QNX Software Systems Limited

libraries (all statically linked), this directory is not needed; however, the `ls` utility is usually linked against the shared `libc`. In such a situation:

```
cd ~ftp
mkdir -m0555 usr
chown root:root usr
mkdir -m0555 usr/lib
chown root:root usr/lib
cd usr/lib
cp /lib/libc.so.3 .
chmod 0555 libc.so.3
chown root:root libc.so.3
ln -s libc.so.3 ldqnx.so.2
```

~ftp/etc

Make this directory owned by the superuser and unwritable by anyone. The `/etc/passwd` and `/etc/group` files must be present for the `LIST` command to be able to produce owner names rather than numbers. The password field in `/etc/passwd` isn't used and shouldn't contain real encrypted passwords. If there's an `/etc/motd` file, its contents are displayed after a successful login. The `/etc/passwd` and `/etc/group` files should be mode 444.

~ftp/pub

Make this directory mode 777 and owned by `ftp`. If any files are to be accessed via the `anonymous` account, the user should place them in this directory.

~ftp/incoming

Make this directory where the anonymous users place files they upload. The owners should be user `ftp` with an appropriate group. Members of this group are the only users with access to these files after they've been uploaded, so these people should know how to deal with them appropriately. To allow anonymous FTP users the ability to see filenames in this directory, set the permissions to 770; otherwise, set the permissions to 370.

Anonymous users are able to upload files to this directory, but they're unable to download them, delete them, or overwrite them due to the umask and disabling of the commands mentioned above.

~ftp/tmp

This directory is used to create temporary files which contain the error messages generated by a conversion or `LIST` command. The owner should be the user `ftp`. The permissions should be 300.

Don't create this directory if you don't want to enable conversion commands or don't want to allow anonymous users uploading files here (see

Copyright © 2014, QNX Software Systems Limited

~ftp/incoming above). Error messages from conversion or LIST commands won't be returned to the user. (This is the traditional behavior.) The /etc/ftpd.conf upload directive can be used to prevent users uploading here.

To set up "ftp-only" accounts to provide FTP only with no valid shell login, you can:

- create a /sbin/nologin file
- copy or link /sbin/nologin to /sbin/ftplogin
- add /sbin/ftplogin to the /etc/shells file

This allows you to log in via FTP into accounts that have /sbin/ftplogin as the login shell.

Based on:

RFC 959, *RFC 1123*, *RFC 2389*, *RFC 2428*

Copyright © 2014, QNX Software Systems Limited

/etc/ftpd.conf

Configuration file for `ftpd`

Name:

`/etc/ftpd.conf`

Description:

The `/etc/ftpd.conf` file specifies various configuration options for *`ftpd`* (p. 850) that apply once a user has authenticated their connection.

Each authenticated user is a member of a *class* (determined by the *`/etc/ftpusers`* (p. 858) file) that associates which entries in this file apply to the user. When parsing entries the following special classes are available:

all

Match any class.

none

Match no class.

The `/etc/ftpd.conf` file consists of a series of lines, each of which may contain a configuration directive, a comment, or a blank line. Directives that appear later in the file override settings by previous directives. This allows "wildcard" entries to define defaults, and then have class-specific overrides.

A "\" is the escape character; it can be used to escape the meaning of the comment character, or if it's the last character on a line, extends a configuration directive across multiple lines. A "#" is the comment character, and all characters from it to the end of line are ignored (unless it's escaped with the escape character).

The *`ftpd`* (p. 850) `STAT` command returns the class settings for the current user as defined by `/etc/ftpd.conf`.

Each configuration line may be one of:

checkportcmd ***class*** **[off]**

Check the `PORT` command for validity. The `PORT` command fails if the IP address specified doesn't match the FTP command connection, or if the remote TCP port number is less than `IPPORT_RESERVED`. It's strongly encouraged that this option be used, espcially for sites concerned with potential security problems with FTP bounce attacks. If *class* is `none`, or if `off` is specified, this feature is disabled.

Copyright © 2014, QNX Software Systems Limited

chroot ***class*** **[*pathformat*]**

Specify the root directory to use with *chroot()* at login. The directory name is created by parsing *pathformat*; the following escape strings may be used:

Escape:	Description:
%c	Class name
%d	Home directory of user
%u	Username
%%	A % character

If *pathformat* isn't specified, or if *class* is none then the default root directory is `/` for REAL users, or the user's home directory for GUEST and CHROOT users.

classtype ***class type***

Set the class type of *class* to *type*, where *type* is one of:

CHROOT

*chroot()*ed users (as per `/etc/ftpchroot` (p. 849)). A *chroot()* is performed after login.

GUEST

Guests (as per the `anonymous` and `ftp` logins). A *chroot()* is performed after login.

REAL

Normal users.

conversion ***class suffix*** **[*type disable command*]**

Define an automatic inline file conversion. If the file to be retrieved ends in *suffix*, and a real file (without a *suffix*) exists, then the output of the command is returned instead of the contents of the file.

suffix

The suffix to initiate the conversion.

type

A list of valid filetypes for the conversion. Valid types are: `f` (file) and `d` (directory).

Copyright © 2014, QNX Software Systems Limited

disable

A file that prevents a conversion if it exists. A filename of "`.`" prevents this action (that is, the conversion is always permitted).

command

Command to run for the conversion. The first word should be the full pathname of the command as *execv()* is used to execute the command. All instances of the word `%s` in the command are replaced with the requested file (without the *suffix*).

Conversion directives specified later on in the file override earlier conversions with the same *suffix*.

display *class* [*file*]

Display the contents of *file* (if it exists) each time the user enters a new directory. Escape sequences are supported; for more information, see the "Display file escape sequences" section in the NetBSD documentation for `ftpd` at *http://netbsd.gw.com/cgi-bin/man-cgi?ftpd++NetBSD-4.0*.

If *file* isn't specified, or *class* is none, disable this.

limit *class count* [*file*]

Limit the maximum number of concurrent connections for *class* to *count*, with `0` indicating unlimited connections. If the limit is exceeded, and *file* is specified, display its contents to the user. This line is ignored if *class* is none or if *count* isn't specified.

homedir *class* [*pathformat*]

Specify the directory to change into at login, and use as the "home" directory of the user for tilde expansion in pathnames, etc. The *pathformat* argument is parsed as per the *chroot* (p. ?) directive.

If *pathformat* isn't specified, or if *class* is none then the default home directory is the home directory of the user for REAL users, or `/` for GUEST and CHROOT users.

maxtimeout *class time*

Set the maximum timeout period that a client may request (default is 2 hours). The period can't be less than 30 seconds, or be equal to the value of the *timeout* (p. ?) directive. This line is ignored if *class* is none or *time* isn't specified.

Copyright © 2014, QNX Software Systems Limited

modify *class* [off]

If *class* is none, or if off is specified, disable these commands: CHMOD, DELE, MKD, RMD, RNFR, and UMASK. Otherwise, enable them.

motd *class* [*file*]

Display the contents of *file* after login as the "message of the day." Escape sequences are supported; for more information, see the "Display file escape sequences" section in the NetBSD documentation for `ftpd` at *http://netbsd.gw.com/cgi-bin/man-cgi?ftpd++NetBSD-4.0*.

If *file* isn't specified, or *class* is none, disable this.

notify *class* [*fileglob*]

Notify the user of any files matching *fileglob*. each time the user enters a new directory,

If *fileglob* isn't specified, or *class* is none, disable this.

passive *class* [off]

If *class* is none, or if off is specified, disallow passive (PASV/LPSV/EPSV) connections.

portrange *class min max*

Set the range of port numbers which are used for the passive data port. The value of *max* must be greater than *min*, and both numbers must be be between `IPPORT_RESERVED` and `IPPORT_ANONMAX`.

rateget *class rate*

Set the maximum get (RETR) transfer rate throttle for *class* to *rate* bytes per second. If *rate* is 0, the throttle is disabled.

An optional suffix may be provided, which changes the intrepretation of *rate* as follows:

- b —Don't modify (optional).
- k —Kilo. Multiply the argument by 1024.
- m —Mega. Multiply the argument by 1048576.
- g —Giga. Multiply the argument by 1073741824.

rateput *class rate*

Set the maximum put (STOR) transfer rate throttle for *class* to *rate* bytes per second. The *rate* argument is parsed as described in *rateget* (p. ?).

Copyright © 2014, QNX Software Systems Limited

template *class* [*refclass*]

Define *refclass* as the template for *class*. All subsequent references to *refclass* in the directives also apply to members of *class*. You'd define a *class* template so that other classes, which share common attributes, can be easily defined without unnecessary duplication. There can be only one template defined at a time. If *refclass* isn't specified, disable the template for *class*.

timeout *class time*

Set the timeout period for inactivity (default is 15 minutes). It can't be less than 30 seconds, or greater than the value for *maxtimeout* (p. ?). This line is ignored if *class* is none or *time* isn't specified.

umask *class umaskval*

Set the umask to *umaskval*. This line is ignored if *class* is none or *umaskval* isn't specified.

upload *class* [off]

If *class* is none, or if off is specified: disable these commands: APPE, STOR, STOU; and modify these: CHMOD, DELE, MKD, RMD, RNFR, UMASK.

Otherwise, enable them.

Default settings

The following defaults are used:

```
checkportcmd  all
classtype     chroot CHROOT
classtype     guest  GUEST
classtype     real   REAL
display       none
limit         all    -1     # unlimited connections
maxtimeout    all    7200   # 2 hours
modify        all
motd          all    motd
notify        none
passive       all
timeout       all    900    # 15 minutes
umask         all    027
upload        all
modify        guest  off
umask         guest  0707
```

Copyright © 2014, QNX Software Systems Limited

/etc/ftpusers

Access-control file for `ftpd`

Name:

`/etc/ftpusers`

Description:

The `/etc/ftpusers` file provides user access control for *`ftpd`* (p. 850) by defining which users may login.

If the `/etc/ftpusers` file **doesn't** exist, all users are **denied** access.

The syntax of each line is:

```
userglob[:groupglob][@host] [directive [class]]
```

where:

userglob

Match against the username. Calls *fnmatch()* (e.g. `f*`).

groupglob

Match against all the groups that the user is a member of. Calls *fnmatch()* (e.g. `*src`).

host

Either a CIDR address (see *inet_net_pton()*) to match against the remote address (e.g. `1.2.3.4/24`), or a glob to match against the remote hostname (e.g. `*.netbsd.org`).

directive

Allow or deny user access.

- `allow` or `yes` — allow user access
- `deny` or `no` — deny user access

If none of the above values are specified, user access is denied.

class

Copyright © 2014, QNX Software Systems Limited

Use this *class* in */etc/ftpd.conf* (p. 853). If *class* isn't specified, it defaults to one of the following:

- `chroot` — there's a match in `/etc/ftpchroot` for the user.
- `guest` — the user name is `anonymous` or `ftp`.
- `real` — neither of the above is true.

No further comparisons are attempted after the first successful match. If no match is found, the user is granted access. This syntax is backward-compatable with the old syntax.

If a user requests a `guest` login, the `ftpd` server checks to see that both `anonymous` and `ftp` have access. If you deny all users by default, you'll need to add both `anonymous allow` and `ftp allow` to `/etc/ftpusers` in order to allow guest logins.

The character:	Meaning:
\	Escape character. It can be used to escape the meaning of the comment character, or if it's the last character on a line, it extends a configuration directive across multiple lines.
#	Comment character. All characters from it to the end of line are ignored (unless it's escaped with the escape character).

Related files

`/etc/ftpchroot`

List of the normal users who should have their session's root directory changed.

Copyright © 2014, QNX Software Systems Limited

fullpath

Display network-qualified pathnames (QNX)

Syntax:

```
fullpath [-v] [name...]
```

Runs on:

QNX Neutrino

Options:

-v

Write verbose output, in this form:

```
name is qualified_path
```

By default, `fullpath` displays only the fully qualified name.

name

A filename.

Description:

The `fullpath` utility resolves all prefixes and symbolic links to display a fully qualified network path for each filename you specify. If no names are given, `fullpath` displays the full pathname of the current working directory.

Examples:

```
$ fullpath -v /usr/bin/CC
/usr/bin/CC is /usr/bin/qcc
```

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 8
G

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “G”.

Copyright © 2014, QNX Software Systems Limited

g++

Compile and link a program (GNU)

We recommend you use *qcc* (p. 1608) instead of invoking g++ directly. You can use the -V option to qcc to invoke g++. For example:

```
qcc -Vgcc_ntoarmv7le my_file.cpp
```

Syntax:

g++_variant [*option* | *filename*]...

where *g++_variant* depends on the target platform, as follows:

Target platform	*g++_variant*
ARMv7	ntoarmv7-g++
x86	ntox86-g++

Runs on:

Linux, Microsoft Windows

Description:

See *gcc* (p. 889) for details.

We recommend you use *qcc or QCC* (p. 1608) instead of gcc to compile and link your programs.

For detailed documentation about gcc, see the GNU website at *http://www.gnu.org/*.

If you have trouble using the default debug format (-gdwarf), switch to -gdwarf-2 or -gstabs.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

/etc/gateways

Specify Internet routing information to `routed`

Name:

`/etc/gateways`

Description:

The `/etc/gateways` file identifies gateways for the `routed` daemon. Typically, the `routed` daemon queries the network and builds routing tables based on the routing information transmitted by other hosts that are directly connected to the network. Gateways that the daemon can't identify through its queries (also called *distant* gateways) may be identified in this file.

When the `routed` daemon starts, it calls this file to:

- find distant gateways which may not be located using only the information from a routing socket
- discover if some of the local gateways are passive
- obtain other parameters (see "*Other parameter settings* (p. 865)").

Gateways must be marked as passive, active or external to indicate how it is to be treated:

active

Is willing to exchange RIP (Routing Information Protocol) packets — they're treated like network interfaces.

passive

Aren't expected to exchange routing information.

external

Are to be considered passive. Another routing process will install such a route if necessary, and other routes to that destination shouldn't be installed by `routed`.

Each entry is contained on a single line. Blank lines and lines starting with a pound sign (#) indicates a comment. An entry may specify user preferences (see "*Other parameter settings* (p. 865)"), or it can indicate whether the route is to a network a specific host using one of the following formats:

```
net Nname[/mask] gateway Gname metric value passive|active|extern
host Hname gateway Gname metric value passive|active|extern
```

Copyright © 2014, QNX Software Systems Limited

Gname

Name or address of the gateway to which RIP (Routing Information Protocol) responses should be forwarded.

Hname* or *Nname

Name of the destination network or host. It may be a symbolic network name (as used in `/etc/hosts` or `/etc/networks`) or an Internet address specified in the conventional "." (dot) notation using the *inet_network()* routine from the internet address manipulation functions, *inet_*()*.

If it's a symbolic network name, then it must either be defined in `/etc/networks` or *`/etc/hosts`* (p. 946), or *`named`* (p. 1332) and must be started before `routed`.

mask

Optional number, between 1 and 32, that indicates the netmask associated with *Nname*.

value

The hop count to the destination host or network.

active

Send RIP responses to the distant active gateway. As long as the gateway is active, information about it is maintained in the internal routing tables, and will be included with any routing information transmitted through RIP. If the gateway doesn't respond for a period of time, the associated route is deleted from the internal routing tables and the RIP responses are advertised via other interfaces. If the distant gateway resumes sending RIP responses, the associated route is restored.

Such gateways can be useful on media that don't support broadcasts or multicasts but otherwise act like classic shared media like Ethernets such as some ATM networks. One can list all RIP routers reachable on the HIPPI or ATM network in *`/etc/gateways`* (p. 863) with a series of "host" lines. Note that it's usually desirable to use RIPv2 in such situations to avoid generating lists of inferred host routes.

passive

Don't exchange RIP (Routing Information Protocol) information. Mark the interface as not to be advertised in updates sent via other interfaces, and turn off all RIP and router discovery through the interface.

Copyright © 2014, QNX Software Systems Limited

Routes through passive gateways are installed in the kernel's routing tables once at startup and aren't included in transmitted RIP responses.

extern

Inform the `routed` daemon that another routing process will install such a route and that alternative routes to that destination shouldn't be installed by `routed`. Information about external gateways is not maintained in the internal routing tables and isn't transmitted through RIP. Such entries are only required when both routers may learn of routes to the same destination.

When debugging is turned on with -T, these lines create pseudo-interfaces. When setting parameters for remote or external interfaces, you should start the lines with: if=alias(*Hname*), or if=remote(*Hname*), etc.

Other parameter settings

Lines that don't start with `net` or `host` must consist of one or more of the following parameter settings, separated by commas or blanks:

bcast_rdisc

Specify that Router Discovery packets should be broadcast instead of multicast.

fake_default=*metric*

Identical effect to the following with the network and mask coming from the specified interface:

-F *net*[/*mask*][=*metric*]

if=ifname

Indicate that the other parameters on the line apply to the interface name `ifname`.

md5_passwd=*XXX*\|*KeyID*[start|stop]

Specify a RIPv2 MD5 password. This keyword is similar to passwd, except that a *KeyID* is required.

no_ag

Turn off collection (aggregation) of subnets in RIPv1 and RIPv2 responses.

no_rdisc

Disable the Internet Router Discovery Protocol.

no_rdisc_adv

Copyright © 2014, QNX Software Systems Limited

Disable the transmission of Router Discovery Advertisements.

no_rip

Disable all RIP processing on the specified interface. If no interfaces are allowed to process RIP packets, `routed` acts purely as a router discovery daemon.

Note that turning off RIP without explicitly turning on router discovery advertisements with rdisc_adv or -s causes `routed` to act as a client router discovery daemon, not advertising.

no_rip_mcast

Cause RIPv2 packets to be broadcast instead of multicast.

no_ripv1_in

Ignore RIPv1 received responses.

no_solicit

Disable the transmission of Router Discovery Solicitations.

no_super_ag

Turn off the collection of networks into supernets in RIPv2 responses.

passwd=*XXX*[|*KeyID*[startlstop]]

Specify a RIPv2 cleartext password that'll be included in all RIPv2 responses sent, and checked in all RIPv2 responses received. Any blanks, tab characters, commas, or #, `|`, or NULL characters in the password must be escaped with a backslash (`\`). The common escape sequences `\n`, `\r`, `\t`, `\b`, and `\xxx` have their usual meanings. The *KeyID* must be unique but is ignored for cleartext passwords. If present, start and stop are timestamps in the form `year/month/day@hour:minute`. They specify when the password is valid. The valid password with the most future is used on output packets, unless all passwords have expired, in which case the password that expired most recently is used, or unless no passwords are valid yet, in which case no password is output. Incoming packets can carry any password that's valid, will be valid within 24 hours, or that was valid within 24 hours. To protect the secrets, the passwd settings are valid only in the `/etc/gateways` file and only when that file is readable only by `root`.

pm_rdisc

Similar to fake_default. When RIPv2 routes are multicast, so that RIPv1 listeners cannot receive them, this feature causes a RIPv1 default route to be broadcast to RIPv1 listeners. Unless modified with fake_default, the

Copyright © 2014, QNX Software Systems Limited

default route is broadcast with a metric of 14. That serves as a "poor man's router discovery" protocol.

rdisc_adv

Specify that Router Discovery Advertisements should be sent, even on point-to-point links, which by default only listen to Router Discovery messages.

rdisc_interval=*N*

Set the nominal interval with which Router Discovery Advertisements are transmitted to *N* seconds and their lifetime to 3**N*.

rdisc_pref=*N*

Set the preference in Router Discovery Advertisements to the optionally signed integer *N* (default preference is 0). Default routes with smaller or more negative preferences are preferred by clients.

redirect_ok

Cause RIP to allow ICMP Redirect messages when the system is acting as a router and forwarding packets. Otherwise, override ICMP Redirect messages.

ripv1_mask=*nname*/*mask1*,*mask2*

Specify that the netmask of the network, of which *nname*/*mask1* is a subnet, should be *mask2*. For example, `ripv1_mask=192.0.2.16/28,27` marks `192.0.2.16/28` as a subnet of `192.0.2.0/27` instead of `192.0.2.0/24`.

It's better to turn on RIPv2 with ripv2_out, instead of using this facility.

ripv2

Enable RIPv2. It's equivalent to `no_ripv1_in` and `no_ripv1_out`.

ripv2_out

Turn on RIPv2 output and cause RIPv2 advertisements to be multicast when possible.

send_solicit

Specify that Router Discovery solicitations should be sent, even on point-to-point links, which by default only listen to Router Discovery messages.

subnet=*nname*[/mask][,*metric*]

Advertise a route to network *nname* with mask *mask* and the supplied metric (default is 1). This parameter must appear by itself on a line. The network

Copyright © 2014, QNX Software Systems Limited

number must specify a full, 32-bit value, as in `192.0.2.0` instead of `192.0.2`.

Although this feature may be useful for filling “holes” in CIDR allocations, it's a dangerous feature and shouldn't be used unless necessary.

trust_gateway=*rname*[|*net1*/*mask1*|*net2*/*mask2*|...]

Cause RIP packets from that router and other routers named in other trust_gateway keywords to be accepted, and packets from other routers to be ignored. If networks are specified, then routes to other networks will be ignored from that router.

Copyright © 2014, QNX Software Systems Limited

gawk

Pattern scanning and processing language (POSIX)

Syntax:

```
gawk [-F ere] [-v var=val] [-W GNU_extension...]
     [--] program [argument]...

gawk [-F ere] -f progfile [-v var=val]
     [-W GNU_extension...] [--] [argument]...
```

Runs on:

QNX Neutrino

Options:

-f *progfile* or --file=progfile

Specifies the name of the gawk program, *progfile*, which contains gawk statements. You can also specify the gawk program as a single argument in the command line by using quotes.

-F *ere* or --field-separator=*ere*

Define the input field separator (FS) to be the extended regular expression *ere*, before any input is read.

-v *var*=*val* or --assign=*var*=*val*

Assign the value *val* to variable *var* before execution of the program begins.

-W *GNU_extension*[, *GNU_extension*]...

Specify a GNU extension. Multiple -W options may be specified, or multiple extensions may be listed in a single -W option if they are separated by commas (or white space if the entire option argument has been quoted from the command line). You can also specify these options using the GNU long options shown in parentheses below.

The GNU extended options are:

compat (--compat)

Run in compatibility mode. In compatibility mode, GNU gawk behaves identically to UNIX awk; none of the GNU-specific extensions are recognized.

Copyright © 2014, QNX Software Systems Limited

copyleft (--copyleft) or copyright (--copyright)

Print the short version of the GNU copyright information message on the error output.

dump-variables[=*file*] (--dump-variables[==*file*])

Print a sorted list of global variables, their types, and final values to *file*. If you don't specify a file, gawk prints this list to the file named `awkvars.out` in the current directory.

exec=*file* (--exec=*file*)

Similar to -f: read the program text from *file*. There are two differences:

- This option terminates option processing; anything else on the command line is passed on directly to the gawk program.
- Command-line variable assignments of the form *var=value* aren't allowed.

gen-po (--gen-po)

Analyzes the source program and generates a GNU gettext Portable Object file on standard output for all string constants that have been marked for translation.

help (--help) or usage (--usage)

Print a relatively short summary of the available options on the error output.

lint[=fatal] (--lint[=fatal]

Provide warnings about constructs that are dubious or nonportable to other awk implementations. If you specify fatal, lint warnings become fatal errors.

lint-old (--lint-old)

Warn about constructs that aren't available in the original version of awk from Version 7 Unix.

non-decimal-data (--non-decimal-data)

Enable automatic interpretation of octal and hexadecimal values in input data.

posix

Copyright © 2014, QNX Software Systems Limited

Turn on compatibility mode, with the following additional restrictions:

- `\x` escape sequences are not recognized.
- The synonym `func` for the keyword `function` is not recognized.
- The operators `**` and `**=` cannot be used in place of `^` and `^=`.

profile[=*file*] (--profile[=*file*])

Enable profiling of `awk` programs. By default, profiles are created in a file named `awkprof.out`. The optional *file* argument lets you specify a different file name for the profile file.

re-interval (--re-interval)

Allow interval expressions in regular expressions.

source=*program-text* (--source=*program-text*)

Use *program-text* as AWK program source code. This option allows the easy intermixing of library functions (used via the -f option) with source code entered on the command line. It is intended primarily for medium to large size AWK programs used in shell scripts. The -W source= form of this option uses the rest of the command-line argument for program-text; no other options to -W are recognized in the same argument.

traditional (--traditional)

Disable all `gawk` extensions.

version (--version)

Print version information for this particular copy of `gawk` on the error output. This is useful mainly for knowing if the current copy of `gawk` on your system is up to date with respect to whatever the Free Software Foundation is distributing.

--

Indicates end of options, in case subsequent item begins with dash (`-`) and is not an option.

program

Copyright © 2014, QNX Software Systems Limited

The text of an awk program that contains awk statements, assuming no -f*progfile* option has been specified. The awk program can either be in a file *progfile* or specified in the command line, taking into account the quoting rules of the shell.

argument

You can intermix either of these two types of arguments:

file

The pathname of a file that contains the input to be read; the input is matched against the set of patterns in the program. If you don't specify any files containing input, or if you specify the dash character (-), the standard input is used.

assignment

You can pass expressions in the form *name*=*value* to gawk, where each instance of *name* represents the name of an gawk variable. Each such variable assignment occurs just prior to the processing of the following *file*, if any. The assignment is done before the *file* argument is executed, and after the `BEGIN` action — if any — of that file.

Description:

The gawk utility executes programs written in the awk programming language; this language specializes in manipulating textual data. An awk program is a sequence of patterns and corresponding actions. When input matching a specified pattern is read, the action associated with that pattern is carried out. The gawk shipped with QNX Neutrino is a port of GNU awk.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

The gawk utility interprets each input line as a sequence of fields, where, by default, a field is a string of nonblank characters. You can change this default white space delimiter by using the builtin variable, *FS* (see *Variables* (p. 876)), or the *-F* (p. ?) *ere* option. The gawk utility denotes the first field in a line $*1*, the second $*2*, and so forth. A $*0* refers to the entire line; setting any other field causes $*0* to be reevaluated.

Each input line matched by the patterns and each input line for the *getline* function (see the section on *Functions* (p. 883)) is limited to 1024 bytes.

Programs in awk are composed of statements of the form:

Copyright © 2014, QNX Software Systems Limited

pattern { *action* }

You can omit either the pattern or the action (that includes the enclosing braces). In the following sections of this description, blank characters between operators and reserved words are ignored, unless otherwise specified. All blank characters are significant inside literal strings and after a function name. A missing pattern matches any line of input, and a missing action is equivalent to an action that writes the matched line of input to standard output.

An awk program follows this general procedure:

1. Starts by executing the actions associated with all BEGIN patterns (BEGIN patterns are described in the section Special patterns — BEGIN and END).
2. Processes each file argument in turn — or standard input if no files are specified — by cyclically reading and saving data from the file until a record separator is seen, evaluating each pattern in the script and executing that action for patterns that evaluate to true (the default record separator is the newline character).
3. Executes the actions associated with all END patterns.

Expressions:

Expressions describe computations used in patterns and actions. Expressions in awk are constructed from such operators as conditionals, logicals, arithmetics, assignments, subscripts, and fields. Expressions take on string or numeric values appropriate to the context. The following table displays valid expressions in groups, starting from the highest priority.

In this table, the abbreviation *expr* represents any expression. The abbreviation *lvalue* represents any entity that you can assign a value to (i.e. on the left side of an assignment operator).

Syntax	Description
(*expr*)	Grouping
$*expr*	References field number *expr*
++*lvalue*	Preincrement *lvalue* by 1
- - *lvalue*	Predecrement *lvalue* by 1
lvalue++	Postincrement *lvalue* by 1
lvalue- -	Postdecrement *lvalue* by 1
expr ^ *expr*	Exponentiation
! *expr*	Logical not
+ *expr*	Unary plus
- *expr*	Unary minus

Copyright © 2014, QNX Software Systems Limited

Syntax	Description
expr * *expr*	Multiplication
expr / *expr*	Division
expr % *expr*	Integer modulus
expr + *expr*	Addition
expr - *expr*	Subtraction
expr expr	String concatenation of two *expr*s
expr < *expr*	Less than
expr <= *expr*	Less than or equal to
expr != *expr*	Not equal to
expr == *expr*	Equal to
expr > *expr*	Greater than
expr >= *expr*	Greater than or equal to
expr1 ~ *expr2*	1 if *expr1* matches the ERE described by *expr2*
expr1 !~*expr2*	1 if *expr2* doesn't match the ERE described by *expr2*
expr `in` *array*	1 if *array*[*expr*] exists
(*index*) `in` *array*	Handles multidimensional arrays
expr && *expr*	Logical AND
expr || *expr*	Logical OR
expr?*expr*:*expr*	Conditional expression — evaluates first *expr* and if nonzero evaluates to the second *expr*; otherwise to the third *expr*
lvalue ^= *expr*	Raise *lvalue* to exponent *expr*
lvalue %= *expr*	Assign *lvalue*%*expr* to *lvalue*
lvalue *= *expr*	Multiply *lvalue* by *expr*
lvalue /= *expr*	Divide *lvalue* by *expr*
lvalue += *expr*	Add *expr* to *lvalue*
lvalue -= *expr*	Subtract *expr* from *lvalue*
lvalue = *expr*	Assign *expr* to *lvalue*

Copyright © 2014, QNX Software Systems Limited

All of the arithmetic operators are based on the C Standard. The conditional expression returns either strings or numbers, depending on the input expressions. Only one of the alternative expressions is evaluated.

In addition to the descriptions in the previous table, an expression can also be a floating-point number, a literal string enclosed by double quotation marks ("), or a variable name.

You can treat a variable or a field as a number or a string at any time, depending on its current usage. There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, you add zero to it. To force an expression to be treated as a string, concatenate the null string ("") to it. Variables and fields are set by the assignment statement:

- *lvalue=expression*

The type of *expression* determines the resulting variable type.

The assignment includes these arithmetic assignments:

+= -= *= /= %= ^= ++ --

each of which produces a numeric result. The left-hand side of an assignment and the target of increment and decrement operators can be one of the following:

- a variable
- an array with index
- a field selector

This is indicated by the following BNF grammar:

***<lvalue>*:**

<*variable*> or | $*expr* or | <*variable*> [<*index*>] or ;

A valid array index consists of one or more comma-separated expressions, with one expression for each dimension of the array. Because awk arrays behave as associative memories, an array index can be any string. Since awk arrays are really one-dimensional, a multidimensional array index is converted to a one-dimensional index by concatenating all expressions, each separated from the other by the value of the *SUBSEP* variable (see *Variables* (p. 876)).

Thus, the following two index operations are equivalent:

- var[*expr1*,*expr2*,... ,*exprn*]
- var[*expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn*]

A multidimensioned index used with the in operator must be parenthesized. The in operator, which tests for the existence of a particular array element, doesn't cause that element to exist. But any other reference to a nonexistent array element automatically creates it.

Copyright © 2014, QNX Software Systems Limited

Comparisons are made numerically if both operands are numeric; otherwise, operands are converted to strings as required and a string comparison is made.

In the table of awk expressions, operators of higher precedence are grouped before those of lower precedence. In expression evaluation, higher precedence operators are evaluated before lower precedence operators. All operators associate to the left except for the assignment operators, the conditional operator (?:), and the exponentiation operator (^). Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, you often need to use parentheses to enforce the proper evaluation precedence.

Variables

You can use variables in an awk program by assigning to them. They don't need to be declared — uninitialized variables have the value of the empty string, which has a numeric value of zero. All variables, including fields, are treated as string variables unless they're used in a clearly numeric context.

Field variables are designated by a $, followed by a number or a numerical expression.

You can create new field variables by assigning a value to them. References to nonexistent fields — i.e. fields after "$(*NF*)" — produce the null string. But, assigning to a nonexistent field (e.g. $(*NF*+2)=5) increases the value of *NF*, creates any intervening fields with the null string as their values, and causes the value of *$0* to be recomputed, with the fields being separated by the value of *OFS*.

This table shows other special variables that gawk sets:

Variable	Meaning
ARGC	The number of elements in the *ARGV* array
ARGV	Array of command-line arguments — excluding options and the *program* argument — numbered from zero to *ARGC*-1
FILENAME	Pathname of the current input file
FNR	Ordinal number of the current record in the current file
FS	Input field separator regular expression; space by default
NF	Number of fields in the current record
NR	Ordinal number of the current record from the start of input

Copyright © 2014, QNX Software Systems Limited

Variable	Meaning
OFMT	Print statement output format for numbers; `"%.6g"` by default
OFS	Print statement output field separation; `space` by default.
ORS	Print statement output record separator; `newline` by default
RLENGTH	Length of string matched by the match function
RS	The first character of the string value of *RS* is the input record separator; `new line` by default. If *RS* is null, records are separated by blank lines, and `newline` is always a field separator, regardless of the value of *FS*
RSTART	Starting position of string matched by *match* function, numbering from 1. This is always equivalent to the return value of the *match* function
SUBSEP	Subscript separator string for multidimensional arrays; the default value is \034

You can modify or add to the arguments in *ARGV*; you can alter *ARGC*. As each input file ends, `gawk` treats the text non-`NULL` element of *ARGV*, up through the current value of *ARGC*-1, as the name of the next input file. Thus, setting an element of *ARGV* to null means that it isn't treated as an input file. A dash (`-`) filename indicates the standard input. If an argument contains an equals sign (=), this argument is treated as an assignment rather than as a *file* argument.

Patterns

The structure of a pattern is specified by the following BNF grammar:

***<pattern>*:**

`BEGIN`

|`END`

|*<simple pattern>*

|*<simple pattern>*, *<simple pattern>*

Copyright © 2014, QNX Software Systems Limited

;

***<simple pattern>*:**

<simple expression>

|/*<ere>*/

;

In other words, a *pattern* is any valid *expression*, or an extended regular expression. In addition, a *pattern* can be a range specified by two of these patterns separated by a comma, or can be one of the two special patterns `BEGIN` or `END`.

Special patterns — BEGIN and END

The gawk utility recognizes two special patterns, `BEGIN` and `END`. `BEGIN` is matched once and its associated action is executed before the first line of input is read and before command-line assignment is done. `END` is matched once and its associated action is executed after the last line of input has been read. These two patterns have associated actions.

`BEGIN` and `END` don't combine with other patterns. Multiple `BEGIN` and `END` patterns are allowed. The actions associated with the `BEGIN` patterns are executed in the order specified in the program, as are the `END` actions. An `END` pattern can precede a `BEGIN` pattern in a program.

If a program consists of:	Then:
Only `BEGIN` blocks	gawk exits without reading its input when the last statement in the `BEGIN` block is executed.
Only `END` blocks or only `BEGIN` and `END` blocks	The input is read before the statements in the `END` block(s) are executed.

Regular expressions

The gawk utility uses the extended expression notation, except that it lets you use C-language conventions for escaping special characters within the extended regular expressions:

Escape	Meaning
`\b`	Backspace
`\f`	Form feed
`\n`	Newline
`\r`	Carriage return

Copyright © 2014, QNX Software Systems Limited

Escape	Meaning
\t	Tab
ddd	1-3 digit octal value *ddd*

If *ere* is an extended regular expression, the pattern:

/ere/

matches any line of input that contains a substring specified by the regular expression. You can limit a regular expression comparison to a specific field or string by using one of the two regular expression matching operators, ~ and !~. For example:

```
$4 ~ /ere/
```

matches any line in which the fourth field matches the regular expression */ere/*.

This pattern:

```
$4 !~ /ere/
```

matches any line in which the fourth field doesn't match the regular expression */ere/*.

You can use an extended regular expression to separate fields by using the -F *ere* option, or by assigning the expression to the builtin variable *FS*. The default field separator is a single space character. The following describes the behavior of *FS*:

1. If *FS* is a single character:
   a. If *FS* is space, skip leading and trailing blanks; fields are delimited by sets of one or more blanks.
   b. If *FS* is any other character *c*, fields are delimited by each single occurrence of *c*.
2. Otherwise, if *FS* is more than one character, *FS* is considered to be an extended regular expression. Each occurrence of the string matching the extended regular expression delimits fields.

Pattern ranges

A pattern range consists of two patterns separated by a comma; in this case, the action is performed for all lines between an occurrence of the first pattern and the following occurrence of the second pattern, inclusive. At this point, the pattern range can be repeated starting at input lines subsequent to the end of the matched range.

Expression patterns

An expression pattern is considered to match — or be true — when the expression evaluates to a nonzero numeric value. Otherwise, the pattern is considered false.

Actions

Copyright © 2014, QNX Software Systems Limited

An action is a sequence of statements. A statement can be one of the statements listed as follows. In this list, optional elements are shown in square brackets ([ ]) and keywords are shown in a `constant-width` typeface.

- `if` (*expression*) *statement* [`else` *statement*]
- `while` (*expression*) *statement*
- `do` *statement* `while` (*expression*)
- `for` ([*expression*];[*expression*]; [*expression*]) *statement*
- `for` (*variable* `in` *array*) *statement*
- `delete` *array*[*index*]
- `break`
- `continue`
- { [*statement*]... }
- *variable* = *expression*
- `next`
- `exit` [*expression*]
- `return` [*expression*]
- `print` [(] [*expression-list*] [)] [*redirection-expression*]
- `printf` [(] *format*[, *expression-list*] [)] [*redirection-expression*]

Any single statement can be replaced by a statement list enclosed in braces (i.e. {}). The statements in a statement list are separated by `newline` characters or semicolons. The symbol # anywhere in a program line — in strings or EREs — begins a comment that is terminated by the end of the line.

Statements are terminated by semicolons or `newline` characters. You can split a long statement across several lines by ending each partial line with a backslash; `newline` characters without backslashes can follow:

- a comma
- an open brace
- a logical AND operator (&&)
- a logical OR operator (||)
- the `do` keyword
- the `else` keyword
- the closing parenthesis of an `if`, `for`, or `while` statement

For example:

```
{ print $1,
        $2 }
```

String constants are surrounded by double quotes ("*string*"). A string expression is created by concatenating constants, variables, field names, array elements, functions, and other expressions.

Copyright © 2014, QNX Software Systems Limited

The *expression* acting as the conditional in an `if` statement is evaluated, and if it is nonzero and nonnull, the next statement is executed. Otherwise, if *else* is present, the statement following the *else* is executed.

The `while`, `do...while`, `for`, `break`, and `continue` statements are based on the C Standard, except in the case of `for (` *variable* `in` *array* `)`, which iterates assigning each *index* of *array* to *variable* in order. The `for` statement has a form that processes each element in an array. The order of processing is unspecified.

The `awk` language supplies arrays that are used for storing numbers or strings. Arrays don't have to be declared, and their sizes change dynamically. The subscripts, or element identifiers, are strings that provide a type of associative array capability. Subscripts can't themselves be arrays.

The `delete` statement removes an individual array element. Thus, the following code deletes an entire array:

```
for (index in array)
    delete array[index]
```

The `next` statement causes all further processing of the current input line to be abandoned.

The `exit` statement invokes all `END` actions in the order in which they occur in the program source. A `next` statement inside an `END` also terminates the program and can optionally set the utility's exit status.

Output statements

Both `print` and `printf` statements send their output to standard output by default. The output is written to the location specified by *redirection-expression*, if one is supplied, as follows:

- >*expression*
- >>*expression*
- |*expression*

In all cases, the *expression* is evaluated to produce a string that's used as a full pathname to write into (for > or >>) or as a command to be executed (for `|`). Using the first two forms, if the file of that name isn't currently open, it is created if necessary, opened, and using the first form, truncated. The output is then appended to the file. Subsequent calls in which *expression* evaluates to the same name simply append output to the file; the file remains open until closed.

The third form writes output onto a stream compatible with *popen()*. If no stream is currently open, the stream is created with the same command name; the stream created is compatible with *popen()* invoked with a mode of `w`. Subsequent calls write output to the existing stream if, in those calls, expression evaluates to the same command name as a stream that's currently open. The stream is closed as if *pclose()* were called with an expression that evaluates to the same command name.

Copyright © 2014, QNX Software Systems Limited

The `print` statement writes the value of each expression argument to the indicated output stream, separated by the current output field separator (see *OFS* in the table of `gawk` variables), and terminated by the output record separator (see *ORS* in the table). String *expressions* are written out as is; numeric *expressions* are written out as if produced by `printf` using a format that is the string value of the variable *OFMT*. The *expression-list* is a comma-separated list of *expressions*. An empty *expression-list* stands for the whole input line ($*0*).

With `printf`, the expressions are printed according to the specified format. A format argument is required — all other arguments in *expression-list* are optional. The string value of the expression *format* is interpreted in a manner similar to the C function *printf()*, as follows. In the *format* string, format specifications begin with the single character %, and can optionally include the following three modifiers:

Modifier	Meaning
-	Left-justify the expression in its field
width	Pad-left to this width as needed; a leading 0 pads with zeros
.prec	Maximum string width, or digits to right of decimal point

Format specifications are terminated by any other character. For each format specification that consumes an argument, the next argument from *expression-list* is evaluated and converted to the appropriate type (string, integer, or floating point). Both `print` and `printf` can output at least 1024 bytes.

The format-specification characters that `gawk` uses are as follows:

Character	Interpretation
c	If the argument is numeric, print a character; if the argument is a string, print only the first character
d	Decimal integer
e	Exponential notation: [-]*d.dddddd*E[+-]*dd*
f	Floating point: [-]*ddd.dddddd*
g	Shorter of `e` or `f` notations; suppress nonsignificant zeros
o	Unsigned octal number
s	String
x	Unsigned hexadecimal number

Copyright © 2014, QNX Software Systems Limited

Character	Interpretation
%	Print a %; no argument consumed

Functions:

The awk language has a variety of builtin functions: arithmetic, string, input/output, and general.

Arithmetic Functions:

The arithmetic functions, except for *int*, are based on the C Standard.

***atan2*(*y*,*x*)**

Return the arctangent of *y*/*x*.

***cos*(*x*)**

Return the cosine of *x*, where *x* is in radians.

***sin*(*x*)**

Return the sine of *x*, where *x* is in radians.

***exp*(*x*)**

Return the exponential function of *x*.

***log*(*x*)**

Return the natural logarithm of *x*.

***sqrt*(*x*)**

Return the square root of *x*.

***int*(*x*)**

Truncate its argument to an integer; truncated toward 0 when *x*>0.

rand()

Return a random number *n*, such that 0<=*n*<1.

***srand*([*expr*])**

Set the seed value for *rand()* to *expr* or to the time of day if *expr* is omitted. The previous seed value is returned.

String functions:

***gsub*(*ere*,*repl*[,*in*])**

Copyright © 2014, QNX Software Systems Limited

Behaves like sub (see below), except that it replaces all occurrences of the regular expression in $*0* or in the *in* argument, when specified.

***index*(*s*, *t*)**

Returns the position, in characters, numbering from 1, in string *s* where string *t* first occurs, or zero if it doesn't occur at all.

***length*[(*s*)]**

Returns the length, in characters, of its argument taken as a string, or of the whole, $*0*, if there is no argument.

***match*(*s*, *ere*)**

Returns the position, in characters, numbering from 1, in string *s* where the extended regular expression *ere* occurs, or zero if it doesn't occur at all. *RSTART* is set to the starting position (which is the same as the returned value), zero if no match is found; *RLENGTH* is set to the length of the matched string, -1 if no match is found.

***split*(*s*, *a*[, *fs*])**

Splits the string *s* into array elements *a*[1], *a*[2],..., *a*[*n*], and returns *n*. The separation is done with the extended regular expression *fs* or with the field separator *FS* if *fs* isn't given.

***sprintf*(*fmt*, *expr*, *expr*, ...)**

Formats the expressions according to the `printf` format given by *fmt* and returns the resulting string.

***sub*(*ere*, *repl*[, *in*])**

Substitutes the string *repl* in place of the first instance of the extended regular expression *ere* in string *in* and returns the number of substitutions. If *in* is omitted, `gawk` uses the current record ($*0*).

***substr*(*s*, *m*[, *n*])**

Returns the *n*-character substring of *s* that begins at position *m*, numbering from 1. If *n* is missing, the length of the substring is limited by the length of the string *s*.

All of the preceding functions that take *ere* as a parameter expect a pattern or a string valued expression that is a regular expression.

Input/Output and general functions:

***close*(*expression*)**

Copyright © 2014, QNX Software Systems Limited

Closes the file or pipe named *expression*.

***expression* | *getline* [*var*]**

Pipes the output of the command string given by expression into *getline*; each successive call to *getline* returns the next line of output from *expression*. This construct has the behavior of *popen()* called with a *mode* of `r`. If *var* isn't specified, $*0* and *NF* are set; if *var* is specified, *var* is set.

getline

Sets $*0* to the next input record from the current input file. This form of *getline* sets *NF*, *NR*, and *FNR* variables.

getline* < *expression

Sets $*0* to the next record from the pathname expression. This form of *getline* sets the *NF* variable.

getline var

Sets variable *var* to the next input record from the current input file. This form of *getline* sets *FNR* and *NR* variables.

getline var* < *expression

Sets *var* from the next record of expression, treated as a pathname. This form of *getline* doesn't set the *NF*, *NR*, and *NFR* variables.

***system* (*expression*)**

Executes the command given by *expression* in a manner consistent with the *system()* function and returns the exit status.

All forms of *getline* return 1 for successful input, zero for end of file, and -1 for an error.

User-defined functions

The awk language also provides user-defined functions. You can define such functions — in the pattern position of a pattern-action statement — as:

- `function` *name*(*args*, ...) {*statements*}

A function can be referred to anywhere in an awk program. In particular, a function call can precede its definition. The scope of a function is global.

Function arguments are passed by value if scalar and by reference if an array name. Argument names are local to the function; all other variable names are global. The number of parameters in the function definition doesn't have to match the number of parameters in the function call. Excess formal parameters can be used as local

Copyright © 2014, QNX Software Systems Limited

variables. If fewer arguments are supplied in a function call than are in the function definition, the extra receiving parameters are left uninitialized.

When you're invoking a function, remember that no white space is allowed between the function name and the opening parenthesis. Function calls can be nested and can be recursive. You can use the `return` statement to return a value.

In the function definition, `newline`s are optional before the opening brace and after the closing brace. Function definitions can appear anywhere in the program where a pattern-action statement is allowed. In a function call, no white space is allowed between the function name and the opening parenthesis that begins the function parameter list.

Sample awk programs:

Note that the following are sample `awk` programs, not complete command lines.

Write to the standard output all input lines for which field 3 is greater than 5:

```
$3 > 5
```

Print every tenth line:

```
(NR % 10) == 0
```

Print any line with a substring that matches the regular expression:

```
/(G|D) (2[0-9][[:alpha:]]*)/
```

Print the second to last field and the last field in each line; separate the fields by a colon:

```
{OFS=":";print $(NF-1), $NF}
```

Print the line number and number of fields in each line. The three strings representing the line number, the colon, and the number of fields are concatenated and the resulting string is written to standard output:

```
{print NR ":" NF}
```

Print lines that are longer than 72 characters:

```
length $0 > 72
```

Print the first two fields in opposite order separated by the *OFS*:

```
{ print $2, $1 }
```

Same as above, with input fields separated by a comma or `space` and `tab` characters, or a combination of all these:

```
BEGIN  {FS = ",[ \t]*|[ \t]+" }
       { print $2, $1 }
```

Add up the first column, and print the sum and the average:

```
    {s += $1 }
END {print "sum is ", s, " average is", s/NR}
```


Copyright © 2014, QNX Software Systems Limited

Print the fields in reverse order, one field per line (i.e. many lines out for each line in):

```
{ for (i = NF; i > 0; --i) print $i }
```

Print all lines between occurrences of the strings start and stop:

```
/start/, /stop/
```

Print all lines whose first field is different from the first field of the previous line:

```
$1 != prev { print; prev = $1 }
```

Simulate echo:

```
BEGIN {
  for (i = 1; i < ARGC; ++i)
    printf "%s%s", ARGV[i], i==ARGC-1?"\n":" "
}
```

If there's a file named `myfile` that contains page headers of the form:

```
Page #
```

and a file named `program` that contains:

```
/Page/{ $2 = n++; }
{ print }
```

then the command line:

```
gawk -f program n=5 myfile
```

would print the file `myfile`, filling in page numbers starting at 5.

See also:

- Dale Dougherty, *sed & awk*, O'Reilly and Associates, 1990
- A.V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988.

Examples:

Print the file `myfile`, which contains page references, filling in page numbers starting at 5:

```
gawk '/Page/{ $2=n++; } { print }' n=5 myfile
```

Files:

By default, input files are text files that are read in order. You can modify either variable *ARGV* or variable *ARGC* to place this default file processing under program control.

The nature of the output files depends on your `awk` program.

Environment variables:

PATH

Copyright © 2014, QNX Software Systems Limited

Defines the search path when looking for commands executed by *system*(*expr*), or input and output pipes.

Exit status:

0

All input files were processed successfully.

>0

An error occurred.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

gcc, g++

Compile and link a program (GNU)

We recommend you use *qcc* (p. 1608) instead of invoking gcc directly.

Syntax:

```
gcc_variant [ option | filename ]...
g++_variant [ option | filename ]...
```

where *gcc_variant* and *g++_variant* depend on the target platform, as follows:

Target platform	*gcc_variant*	*g++_variant*
ARMv7	`ntoarmv7-gcc`	`ntoarmv7-g++`
x86	`ntox86-gcc`	`ntox86-g++`

Runs on:

Linux, Microsoft Windows

Description:

We recommend you use *qcc* or *QCC* (p. 1608) instead of gcc to compile and link your programs. You can use the -V option to qcc to invoke gcc. For example:

```
qcc -Vgcc_ntoarmv7le my_file.c
```

For detailed documentation about gcc, see *http://gcc.gnu.org/onlinedocs/gcc-4.7.3/gcc/*.

- If you use exceptions, you must link with the -lang-c++ option to *qcc* (p. 1608). This option is the default for QCC.
- Even with exceptions disabled, the default *new()* operator throws a `std::out_of_memory` exception if there isn't enough memory. If you want *new()* to return NULL instead of throwing an exception, overload the *new()* operator with your own.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

gcov

Gather code coverage data (GNU)

Syntax:

gcov_variant [*options*] *sourcefile*

where *gcov_variant* depends on the target platform, as follows:

Target platform	*gcov_variant*
ARMv7	`ntoarmv7-gcov`
x86	`ntox86-gcov`

Runs on:

Linux, Microsoft Windows

Options:

-b or --branch-probabilities

Write branch frequencies to the output file and branch summary info to standard output. This lets you see how often each branch was taken.

-c or --branch-counts

Write branch frequencies as the number of branches taken instead of the percentage of branches taken.

-f or --function-summaries

Output summaries for each function in addition to the file level summary.

-h or --help

Display `gcov` command-line options and exit.

-l or --long-file-names

Create long file names for included source files.

For example, if `x.h` was included in `a.c`, then running `gcov -l` on `a.c` will produce `a.c##x.h.gcov` instead of `x.h.gcov`. This can be useful if `x.h` is included in multiple source files.

-n or --no-output

Copyright © 2014, QNX Software Systems Limited

Don't create the output file.

-o *directory|file* or --object-directory *directory*, --object-file *file*

Specify the directory containing the gcov data files or the object path name.

The gcov utility searches for the `.bb`, `.bbg`, and `.da` data files using this option. If a *directory* is specified, the data files are in that directory and named after the source file name without its extension. If a *file* is specified, the data files are named after that file without its extension. If this option isn't specified, it defaults to the current directory.

-p or --preserve-paths

Preserve complete path information in the names of generated `.gcov` files.

Without -p only the filename component is used. With -p, all directories are used, with `/` characters translated to # characters, `.` directory components removed and `..` components renamed to ^. This is useful if source files are in several different directories.

The -p option also affects the -l option.

-v or --version

Display the gcov version number on the standard output and exit.

Description:

The gcov utility produces code coverage data for an application compiled with the -Wc,-fprofile-argcs -Wc,-ftest-coverage options to *qcc* (p. 1608) (or the -fprofile-arcs -ftest-coverage options to *gcc* (p. 889)). This data is interpreted visually for you by the Code Coverage perspective in the Integrated Development Environment (IDE).

For detailed documentation about gcov, see the gcc documentation on the GNU website at *http://www.gnu.org/*.

Exit status:

0

Success.

not 0

An error occurred.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

gdb

Debugger (GNU)

Syntax:

gdb_variant [*options*] [*executable*] [*core_file* | *pid*]

where *gdb_variant* depends on the target platform, as follows:

Target platform	*gdb_variant*
ARMv7	`ntoarmv7-gdb`
x86	`ntox86-gdb`

Runs on:

Linux, Microsoft Windows

Options:

-[no]async

Enable (disable) asynchronous version of CLI.

-b *bps*

Set the line speed (baud rate or bits per second) of any serial interface used by gdb for remote debugging.

-batch

Run in batch mode. Exit with status 0 after processing all the command files specified with -x (and all commands from initialization files, if not inhibited with -n). Exit with nonzero status if an error occurs in executing the gdb commands in the command files.

Batch mode may be useful for running gdb as a filter, for example to download and run a program on another computer. To make this more useful, the message:

```
Program exited normally.
```

(which is ordinarily issued whenever a program running under gdb control terminates) isn't issued when running in batch mode.

-cd=*directory*

Copyright © 2014, QNX Software Systems Limited

Run gdb using *directory* as its working directory, instead of the current directory.

-command=*file*

Execute gdb commands from *file*. See "Command files" in the full online GNU documentation.

-core=*file*

Examine *file* as a core dump.

-dbx

DBX compatibility mode.

-directory=*directory*

Add *directory* to the path to search for source files.

-epoch

Output information used by epoch emacs-GDB interface.

-exec=*file*

Use *file* as the executable file to execute when appropriate, and for examining pure data in conjunction with a core dump.

-fullname

GNU Emacs sets this option when it runs gdb as a subprocess. It tells gdb to output the full filename and line number in a standard, recognizable fashion each time a stack frame is displayed (which includes each time your program stops). This recognizable format looks like two `\032` characters, followed by the file name, line number and character position separated by colons, and a newline. The Emacs-to-gdb interface program uses the two `\032` characters as a signal to display the source code for the frame.

-help

Display all available options and briefly describe their use.

-interpreter=*file*

Select a specific interpreter or user interface.

-mapped

Use mapped symbol files if supported on this system.

Copyright © 2014, QNX Software Systems Limited

This option depends on operating system facilities that aren't supported on all systems.

If memory-mapped files are available on your system through the *mmap()* system call, you can use this option to have `gdb` write the symbols from your program into a reusable file in the current directory.

For example, if the program you're debugging is called `/tmp/fred`, the mapped symbol file is `./fred.syms`. Future `gdb` debugging sessions notice the presence of this file, and can quickly map in symbol information from it, rather than read the symbol table from the executable program.

The `.syms` file is specific to the host machine where `gdb` is run. It holds an exact image of the internal `gdb` symbol table. It can't be shared across multiple host platforms.

-nw

Do not use a window interface.

-nx

Don't execute commands from any initialization files (normally called `.gdbinit`). If you don't specify this option, these `.gdbinit` files are executed before any command-line options and arguments have been processed.

For more information on initialization files, see “Command files” in the full online GNU documentation.

-quiet

Don't print the introductory and copyright messages. These messages are also suppressed in batch mode.

-readnow

Read each symbol file's entire symbol table immediately, rather than read it incrementally as needed. This makes startup slower, but makes future operations faster.

The -mapped and -readnow options are typically combined in order to build a `.syms` file that contains complete symbol information. (For more information, see “Commands to specify files” in the full online GNU documentation.)

Here's how you can build a `.syms` file for future use:

```
gdb -batch -nx -mapped -readnow programname
```


Copyright © 2014, QNX Software Systems Limited

-se=*file*

Read the symbol table from *file* and use it as the executable file.

-symbols=*file*

Read the symbol table from the file *file*.

-tty=*device*

Run using *device* for your program's standard input and output.

-version

Print version information and then exit.

-w

Use a window interface.

-write

Set writing into executable and core files.

-xdb

XDB compatibility mode.

Description:

Invoke the GNU Debugger by running `gdb`. Once started, `gdb` reads commands from the terminal until you tell it to exit.

If you start `gdb` from the command line on a QNX Neutrino system, `gdb` sets the ***LD_BIND_NOW*** to 1 to force all binding to be done immediately instead of lazily. For more information, see "Optimizing the runtime linker" in the Compiling and Debugging chapter of the QNX Neutrino *Programmer's Guide*.

You can abbreviate a `gdb` command to the first few letters of the command name, if that abbreviation is unambiguous; and you can repeat certain `gdb` commands by typing just **Enter**. You can also use the **Tab** key to get `gdb` to fill out the rest of a word in a command (or to show you the alternatives available, if there's more than one possibility).

You can also run `gdb` with a variety of arguments and options, to specify more of your debugging environment at the outset.

Unless you specify the -nx option, this utility runs commands in an initialization file before running any command-line options. This file may be specific to the *gdb_variant* invoked, for instance, if you invoke `ntoarmv7-gdb`, the utility runs the commands in the `${HOME}/.ntoarmv7-gdbinit` file. If no such *gdb_variant* initialization file

Copyright © 2014, QNX Software Systems Limited

exists, the utility runs commands found in the generic `${HOME}/.gdbinit` file instead. If you specify -command, *file* overrides these default files.

The command-line options described here are designed to cover a variety of situations; in some environments, some of these options might not work.

You can start with both an executable program and a core file specified:

```
gdb program core
```

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

getconf

Get system configuration values (POSIX)

Syntax:

```
getconf system_var
getconf path_var pathname
```

Runs on:

QNX Neutrino

Options:

None.

Description:

This utility gets system configuration values, including:

- configuration strings, whose names start with _CS_
- configurable system limits, whose names start with _SC_
- configurable limits associated with a path, whose names start with _PC_

The names of these variables are in uppercase, but `getconf` and *setconf* (p. 1745) let you use any case, omit the leading underscore, or the entire prefix — provided that the rest of the name is unambiguous.

The first form writes to standard output the value of the specified system variable. The possible values of *system_var* are those for *sysconf()* and *confstr()* (see the *C Library Reference*):

_CS_ARCHITECTURE

The name of the instruction set architecture for this node's CPU(s).

_CS_DOMAIN

The domain name.

_CS_HOSTNAME

The name of this node in the network.

Copyright © 2014, QNX Software Systems Limited

A hostname can consist only of letters, numbers, and hyphens, and must not start or end with a hyphen. For more information, see *RFC 952*.

_CS_HW_PROVIDER

The name of the hardware manufacturer.

_CS_HW_SERIAL

Serial number associated with the hardware.

_CS_LIBPATH

A value similar to the ***LD_LIBRARY_PATH*** environment variable that finds all standard libraries.

_CS_LOCALE

The name of the current locale.

_CS_MACHINE

This node's hardware type.

_CS_PATH

A value similar to the ***PATH*** environment variable that finds all standard utilities.

_CS_RELEASE

The current OS release level.

_CS_RESOLVE

The contents of the `resolv.conf` file, excluding the domain name.

_CS_SRPC_DOMAIN

The secure RPC domain.

_CS_SYSNAME

The name of the operating system.

_CS_TIMEZONE

Time zone string (***TZ*** style)

_CS_VERSION

The current OS version number.

Copyright © 2014, QNX Software Systems Limited

_SC_AIO_PRIO_DELTA_MAX

The maximum amount by which a process can decrease its asynchronous I/O priority level from its own scheduling priority.

_SC_ARG_MAX

Maximum length of arguments for the *exec*()* functions, in bytes, including environment data.

_SC_CHILD_MAX

Maximum number of simultaneous processes per real user ID.

_SC_CLK_TCK

The number of intervals per second used to express the value in type `clock_t`.

_SC_DELAYTIMER_MAX

The maximum number of times a timer can overrun and you can still detect it.

_SC_GETGR_R_SIZE_MAX

If defined (not -1), the maximum size of buffer that you need to supply to *getgrgid_r()* for any memory that it needs to allocate.

_SC_GETPW_R_SIZE_MAX

If defined (not -1), the maximum size of buffer that you need to supply to *getpwent_r()*, *getpwuid_r()*, *getspent_r()*, or *getspnam_r()* for any memory that they need to allocate.

_SC_JOB_CONTROL

If this variable is defined, then job control is supported.

_SC_NGROUPS_MAX

The maximum number of simultaneous supplementary group IDs per process.

_SC_OPEN_MAX

Maximum number of files that one process can have open at any given time.

_SC_PAGESIZE

The default size of a thread's guard area.

_SC_SAVED_IDS

Copyright © 2014, QNX Software Systems Limited

If this variable is defined, then each process has a saved set-user ID and a saved set-group ID.

_SC_SEM_NSEMS_MAX

The maximum number of semaphores that one process can have open at a time. The `getconf` utility reports a value of -1 to indicate that this limit is indeterminate because it applies to both named and unnamed semaphores. The kernel allows an arbitrary number of unnamed semaphores (they're kernel synchronization objects, so the number of them is limited only by the amount of available kernel memory).

_SC_SIGQUEUE_MAX

The maximum number of outstanding realtime signals sent per process.

_SC_THREAD_STACK_MIN

The minimum stack size for a thread.

_SC_TZNAME_MAX

The maximum length of the names for time zones.

_SC_VERSION

The current POSIX version that is currently supported. A value of 198808L indicates the August (08) 1988 standard, as approved by the IEEE Standards Board.

The second form writes to standard output the value of the specified path variable for the given path. The possible values of *path_var* are those for *pathconf()* (see the *C Library Reference*):

_PC_ASYNC_IO

Defined if asynchronous I/O is supported for the file.

_PC_CHOWN_RESTRICTED

If defined (not -1), indicates that the use of the *chown* (p. 129) function is restricted to a process with `root` privileges, and to changing the group ID of a file to the effective group ID of the process or to one of its supplementary group IDs.

_PC_LINK_DIR

Defined (not -1) if the filesystem permits the unlinking of a directory.

_PC_LINK_MAX

Maximum value of a file's link count.

Copyright © 2014, QNX Software Systems Limited

_PC_MAX_CANON

Maximum number of bytes in a terminal's canonical input buffer (edit buffer).

_PC_MAX_INPUT

Maximum number of bytes in a terminal's raw input buffer.

_PC_NAME_MAX

Maximum number of bytes in a file name (not including the terminating null).

_PC_NO_TRUNC

If defined (not -1), indicates that the use of pathname components longer than the value given by _PC_NAME_MAX will generate an error.

_PC_PATH_MAX

Maximum number of bytes in a pathname (not including the terminating null).

_PC_PIPE_BUF

Maximum number of bytes that can be written atomically when writing to a pipe.

_PC_PRIO_IO

Defined (not -1) if prioritized I/O is supported for the file.

_PC_SYNC_IO

Defined (not -1) if synchronous I/O is supported for the file.

_PC_VDISABLE

If defined (not -1), this is the character value which can be used to individually disable special control characters in the `termios` control structure.

Examples:

```
$ getconf _CS_PATH
/bin
$ getconf _SC_ARG_MAX
61440
$ getconf _PC_LINK_MAX /bin
65535
```

Environment variables:

LANG

Copyright © 2014, QNX Software Systems Limited

The locale to use for the locale categories.

QNX Neutrino currently supports only the POSIX (i.e. C) locale.

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

getfacl

Get the access control list (ACL) for files or directories

Syntax:

```
getfacl [-q] [path ...]
```

Runs on:

QNX Neutrino

Options:

-q

Be quiet; don't write commented information about the file or directory name and ownership. This is useful when you're dealing with paths with unprintable characters.

***path* ...**

The file or directory that you want to get the ACL for. If you specify a hyphen (`-`) or don't specify any paths, `getfacl` reads them, one per line, from standard input until you press **Ctrl**–**D**.

Description:

The `getfacl` utility gets the access control list for files or directories. For an overview of ACLs, see "Access Control Lists (ACLs)" in the QNX Neutrino *User's Guide*.

Examples:

Get the ACL for `my_file`:

```
# getfacl my_file
# file: my_file
# owner: mabel
# group: docs
user::rwx
group::rwx
group:techies:r-x
mask::rwx
other::---
```

Get the ACL, but suppress the comments:

```
# getfacl -q my_file
user::rwx
group::rwx
group:techies:r-x
```

Copyright © 2014, QNX Software Systems Limited

```
mask::rwx
other::---
```

Exit status:

0

Success.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

getty

Set terminal modes for system access (NetBSD)

Syntax:

```
getty
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `getty` utility sets the terminal mode.

Files:

`/etc/config/gettytab`, `/etc/config/gettytab.${HOSTNAME}`

Terminal configuration database.

Copyright © 2014, QNX Software Systems Limited

gns

Advertise, look up, and use (connect to) a service across a network

Syntax:

```
gns [-cv] [nodename ...]
gns [-sv] [nodename]
```

Runs on:

QNX Neutrino

Options:

-c

Run in client mode. This is the default.

-s

Run in server mode.

-v

Be verbose.

nodename

A node running a GNS server.

Description:

The global name service (gns) manager is a standalone resource manager. With the help of gns, an application could advertise, look up, and use (connect to) a service across network, without knowing the detail of where the service is, or who is the provider.

GNS runs in two different modes: server and client. A server-mode manager is a central database that stores advertised services, and handles lookup and connect requests. A client-mode manager relays advertise, lookup, and connect requests between a local application and the gns server.

APIs and advertising rules

There are several functions in the *C Library Reference* that you can use: *name_attach()* to advertise, *name_open()* to connect to, *name_close()* to disconnect from a certain service, and *name_detach()* to remove a name from the namespace.

Copyright © 2014, QNX Software Systems Limited

An application advertises (attaches) a service (i.e. represented by a path name) either *locally* or *globally*. If an application attaches a service locally, then applications from another machine can't look up this service through the `gns` utility. If an application attaches its services globally, then any machine that's on the network and is running the `gns` manager can access the services.

An application can attach a service locally, but only if there isn't another application that's attached locally to the same service. There's no credential restriction for applications that are attached as local services.

An application can attach a service globally only if the application has `root` privileges.

Even though attaching globally is network-wide, an application on another machine could still attach globally with the same service name. This allows service redundancy, since the same service is available from multiple hosts on the network.

Although we have used the *name_** API for the implementation of GNS, there is slight behavior change with respect to the previous implementation of the QNX Neutrino RTOS.

Before, when an application called *name_open()* to connect to a service, the server was not aware of that. This has been changed now —an `_IO_CONNECT`/`_IO_OPEN` message is sent to the server.

The server application has to be modified to handle a possible `_IO_CONNECT` message coming in. For an example, see the documentation for *name_attach()* in the QNX Neutrino *C Library Reference*.

Path namespace

A service is represented by a path namespace (without a leading "/") and is registered under `/dev/name/global` or `/dev/name/local`, depending on how it attaches itself.

Every machine running a `gns` client or server on the same network has the same view of the `/dev/name/global` namespace. Each machine also has a `/dev/name/local` namespace that's local to each machine, and therefore, is different.

For details, see the Examples section.

Connection rules for GNS

Applications that wish to connect to a global name service can use the *name_open()* API. If the same service is registered by multiple hosts, the rules that determine which specific instance of the service you connect to are:

- If a service provider exists in the same machine as the application that requested the service, the `gns` manager tries to connect to the local service provider first. If

Copyright © 2014, QNX Software Systems Limited

the connection succeeds, the application communicates with its service provider locally, to gain better performance.

- If there's no local service provider, or, for some reason, the local service provider refuses to provide the service, the `gns` manager tries to connect to other providers. If there are multiple remote service providers, the order of trying them (i.e. who gets the connection first) isn't defined.

Multiple GNS servers

It's possible to start multiple global name service managers in server mode on different machines.

Multiple service domains

You can set some clients to connect to `server1`, and some clients to `server2`. This creates separate "service domains," where clients connected to `server1` (in "service domain1") can't use services registered on `server2` (in "service domain2").

On some clients:

```
gns -c server1
```

On others:

```
gns -c server2
```

Redundant GNS servers

Since the GNS server is a central database, the loss of redundant GNS servers could mean the interruption of service (advertise, lookup, use) across the network. To solve this problem, you can start multiple GNS servers and point clients to all of them.

On server1:

```
gns -s
```

On server2:

```
gns -s
```

On all clients:

```
gns -c server1 server2
```

In this case, every time an application tries to register a global service, the registration is sent to both `server1` and `server2`. Every time an application tries to connect to a global service, the request is attempted on both `server1` and `server2`.

The GNS on `server1` doesn't communicate with the GNS on `server2`. This means that if an application on `server1` wants to register a global service on client nodes, the `gns` process on `server1` can't forward the request to `server2`, because a `gns` process can't act as a client and server at the same

Copyright © 2014, QNX Software Systems Limited

time. This however, doesn't affect the applications that try to connect to that service because both servers are attempted.

Auto-scanning client

Each GNS manager is registered under `/dev/name/gns_server` or `/dev/name/gns_client`. If you start a GNS client without specifically assigning a target server, it performs an auto-scan to find the GNS server(s) on the local network.

On a client:

```
gns -c
```

An auto-scan is performed by going through the local network directory (usually `/net`), and trying to see if any machine has a pathname of `/net/`*machine*`/proc/mount/dev/name/gns_server`.

An auto-scan isn't *guaranteed* to find a server. This is because the Qnet network isn't guaranteed to have all local machines under `/net`.

The benefit of an auto-scan client lies in the event of losing the connection to a server. The client will rescan to find another server when this happens. This makes it easy to start another GNS server, synchronize it with the first one (we'll discuss synchronizing later), and then kill the first GNS server.

If a client starts specifying specific GNS server(s) on the command line, it won't perform an auto-scan. If it loses the connection to its server, it tries to reestablish the connection every time a registration, lookup, or connect request is made.

Backup server mode

A GNS manager can be started in a "backup server" mode. Simply start a GNS manager in server mode, and pass a specific server machine on the command line.

On node1, start a normal server:

```
gns -s
```

and, on node2, start a backup server:

```
gns -s node1
```

The GNS manager on `node2` synchronizes with the GNS manager on `node1`, gets all the global service information from `node1`, and stores it locally.

All GNS clients that are already connected to the GNS server on `node1` are notified about the new server on `node2`, and they connect to it. These clients then have multiple servers, as if they had been started as follows:

```
gns -c node1 node2
```

Copyright © 2014, QNX Software Systems Limited

GNS and tightly coupled network

For a tightly coupled network, you may start the GNS manager in server mode only. All client nodes could prefix the server's name space, instead of running a GNS manager in client mode locally.

On a server:

```
gns -s
```

On other clients:

```
ln -sP /net/gsrv/dev/name/global /dev/name/global
```

When a *service provider* registers its global service name from a client node (not running `gns -c`), the information is directly sent to the GNS manager on server. Subsequently, if a *service consumer* tries to look up the service name, the lookup message is sent to the GNS manager on server, which then returns the service provider information.

If a service provider registers a local service name, it simply registers itself in the local path manager. The local path manager serves a service consumer when it starts to look up the local service.

You must take note of the differences of running the GNS manager in client mode rather than using the prefixes described above. A client-mode GNS manager running locally may cache some service provider information, such that every lookup does not need to go to the GNS manager on the server. This ensures that a lookup usually succeeds even if there is a period of lost connection with the GNS manager.

Also, if the path namespace is prefixed, the client can't look at the `/dev/name/net` directory to tell which node provides which services. This information is, however, not important in normal advertise/lookup operations.

Special pathname

All GNS managers are registered as `/dev/name/gns_server` or `/dev/name/gns_client`. This pathname can provide some statistical information about the GNS manager. For example:

```
$ cat /dev/name/gns_client
Global Name Service Mode:    Client
                  Server:    xtang (connected)

Registered services:
   net/netsrv.ott.qnx.com/0,1818649,1,0,-1     (No Expiration)
   network/tcpip/51,20533309,1,0,12            (Fri Feb  7 13:57:39 2003)
   printer/ps/techpub/0,1826845,1,0,12         (No Expiration)
```

From the above, we infer that the `gns` process is a client, it has only one server (the machine named `xtang`), and it's already connected to the server. This also lists all the services known to the GNS manager. Note that the `network/tcpip` service has an expiration time, suggesting that this is a cached entry resulting from querying the `gns` server.

Copyright © 2014, QNX Software Systems Limited

Examples:

A typical network with `gns` looks like this:

Server node:

```
gns -s
```

Client(s) node:

```
gns -c server1 server2
```

Here's an example after a service called `printer/ps/techpub` has attached itself globally:

```
$ ls -l /dev/name/global/
total 2
dr-xr-xr-x  0 root      techies           1 Feb 06 16:20 net
dr-xr-xr-x  0 root      techies           1 Feb 06 16:21 printer

$ ls -l /dev/name/global/printer/ps
total 1
dr-xr-xr-x  0 root      techies           1 Feb 06 16:21 techpub
```

Notice how `/dev/name/global/printer/ps/techpub` is registered. The path `/dev/name/global/net` is reserved by the `gns` manager (therefore, an application can't attach a service started as `net/`). The machines under `/dev/name/global/net` represent machines that run the GNS manager. For example, the following listing shows that there are two machines running the GNS manager:

```
$ ls -l /dev/name/global/net total 2
dr-xr-xr-x  0 root      techies           1 Feb 06 18:32 netsrv.ott.qnx.com
dr-xr-xr-x  0 root      techies           2 Feb 06 16:20 xtang.ott.qnx.com
```

Multiple application servers on different machines can attach to the same service. Therefore one pathname could represent multiple servers. In order to show that, the GNS manager automatically creates a name in the format *nd pid chid handle file_type* under the registered pathname, as follows:

```
$ ls /dev/name/global/printer/ps/techpub
0,16834613,1,0,12
8,1826845,1,0,12
```

From the above, you conclude that there are two applications attached to the `printer/ps/techpub` service.

Sometimes, you may wish determine what service machine `netsrv.ott.qnx.com` provides. To find that out, you can look into `/dev/name/global/net/netsrv.ott.qnx.com` — it shows the services attached by a process on `netsrv.ott.qnx.com`:

```
$ ls /dev/name/global/net/netsrv.ott.qnx.com/printer/ps/techpub
0,1826845,1,0,12
```

To know exactly which machine registered a service, try this:

```
$ ls -l /dev/name/global/printer/ps/techpub
total 0
lr-xr-xr-x  0 root      techies           0 Feb 06 16:48
0,16834613,1,0,12 ->
../../../net/xtang.ott.qnx.com/printer/ps/techpub/0,16834613,1,0,12
lr-xr-xr-x  0 root      root              0 Feb 06 16:28 8,1826845,1,0,12
```

Copyright © 2014, QNX Software Systems Limited

```
->
../../../net/netsrv.ott.qnx.com/printer/ps/techpub/0,1826845,1,0,12
```

From the above, you conclude that a process on machine `xtang.ott.qnx.com` with process ID 16834613 has registered the `printer/ps/techpub`service; meanwhile, process 1826845 on machine `netsrv.ott.qnx.com` also registered the same service.

Copyright © 2014, QNX Software Systems Limited

gprof

Code profiler (GNU)

Syntax:

gprof_variant options [*executable* [*data-files...*]] [> *outfile*]

where *gprof_variant* depends on the target platform, as follows:

Target platform	*gprof_variant*
ARMv7	`ntoarmv7-gprof`
x86	`ntox86-gprof`

Runs on:

Linux, Microsoft Windows

Description:

The `gprof` utility produces code-profiling data for an application compiled with the -p option to *qcc* (p. 1608) (or the -pg option to *gcc* (p. 889)). This data is interpreted visually for you by the Application Profiler perspective in the Integrated Development Environment (IDE).

For detailed documentation about `gprof`, see the the GNU website at *http://www.gnu.org/*.

Exit status:

0

Success.

not 0

An error occurred.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

grep

Search for string patterns (POSIX)

Syntax:

```
grep [-E|-F] [-chilnqsvx]
     [-e expression | -f expression_file]...
     [file...]
grep [-E|-F] [-chilnqsvx] expression [file...]
```

Historical UNIX versions:

```
egrep [-cilnqsvx]
      [-e expression | -f expression_file]...
      [file...]
egrep [-cilnqsvx] expression [file...]

fgrep [-cilnqsvx]
      [-e expression | -f expression_file]...
      [file...]
fgrep [-cilnqsvx] expression [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-c

Write only a count of selected lines to standard output.

-E

Use extended regular expression (ERE) syntax.

-e *expression*

A regular expression, of type determined by the -E, -F options. You can use more than one -e option when you need to specify more than one expression.

-F

Treat *expression* as a fixed string instead of a regular expression. (Search for a fixed string or fixed strings.)

-f *expression_file*

File containing a set of regular expressions, each separated by a newline. The type of the expressions is determined by the -E and -F options. This

Copyright © 2014, QNX Software Systems Limited

form is used when more than one expression needs to be specified. You can specify more than one -f option.

-h

(QNX Neutrino extension to `grep` only) Don't prefix matched lines with a filename. This option applies only when you invoke `grep` with more than one file to search.

-i

Ignore uppercase and lowercase distinctions during comparisons.

-l

("el") Write only the names of files containing selected lines to standard output.

-n

Before each output line, display the line's line number in the file.

-q

Be quiet; don't write anything to the standard output, regardless of matching lines.

-s

Suppress the error messages ordinarily written for nonexistent or unreadable files. Other error messages aren't suppressed.

-v

Select only those lines that don't match the specified patterns.

-x

Consider as matching lines only input lines selected against an entire fixed string or regular expression.

expression

A regular expression, whose type is determined by the -E and -F options. This form is used when only one expression is specified on the command line. Any names specified after this option are treated as input files.

file

The text file to be input. The default is standard input.

Copyright © 2014, QNX Software Systems Limited

Description:

The `grep` utility searches input for lines matching the expression(s) given. When an input line matches any of the expressions, it is said to be "selected." By default, selected lines are written to standard output.

Numerous options allow variations upon the output format. For example, to reverse the meaning of the output, the -v option could be used.

There are three types of regular expressions understood by `grep`: basic, extended, and fixed. If you don't specify -E or -F, the expression(s) are taken to be basic regular expressions.

Basic and extended regular expressions are similar to arithmetic expressions in that larger expressions are formed by combining smaller expressions and operators according to some precedence rule.

Regular expressions have an "invisible" operator, i.e. concatenation. The concatenation of two expressions means match the one on the left, then the one on the right.

The smallest expression is a single character.

Basic regular expressions

The following table summarizes the Basic Regular Expressions (BRE), and the precedence of the operators:

Expression	Meaning
`\(` *expression* `\)`	Subexpression. Match the pattern *expression*. Used for back references (see below), and precedence
`\`*N*	Back-reference. Match the exact string that the *N*th subexpression did
`.`	(Dot) match any single character
[*charset*]	Match any member of the set *charset* (see below)
c	Match any nonspecial character
`\`*c*	Match literal *c*. The character may not be `(`, `)`, `{`, `}`, or any digit from `1` through `9`. The `\` is usually used to escape `*`, `$`, `^`, `.`, `[` and `]`. `\\` matches a literal "`\`". `\` has no special meaning inside a bracket expression.
limited_expression`*`	Match any number of repetitions of *limited_expression* including zero.

Copyright © 2014, QNX Software Systems Limited

Expression	Meaning
limited_expression\{*M*\}	Match exactly *M* repetitions of *limited_expression*
limited_expression\{,*N*\}	Match zero to *N* repetitions of *limited_expression*
limited_expression\{*M*,*N*\}	Match *M* to *N* repetitions of *limited_expression*
expr0expr1	(Concatenation) match *expr0* then *expr1*
^*expression*	Match *expression* only at beginning of line
expression$	Match *expression* only at end of line

A *limited_expression* is restricted to a a back-reference, a subexpression, or a BRE matching a single character.

A *charset* is formed by concatenation of the following operators:

Expression	Meaning
c	Any character *c*
c-*d*	Any character in the range from *c* to *d*
[:alpha:]	Any alphabetic character
[:upper:]	Any uppercase character
[:lower:]	Any lowercase character
[:digit:]	Any numeric character
[:alnum:]	Any numeric or alphabetic character
[:xdigit:]	Any character used to represent a hexadecimal number
[:space:]	Any character that is a whitespace
[:print:]	Any printable character
[:punct:]	Any character that is punctuation
[:graph:]	Any character with a graphic representation
[:cntrl:]	Any character used for control

If the *charset* begins with the caret (^), the set is inverted. For example:

```
[^[:alpha:]]
```

Copyright © 2014, QNX Software Systems Limited

means match any nonalphabetic character. (This can also be expressed by `[^a-zA-Z]`.)

Extended Regular Expressions

The Extended Regular Expressions (ERE) are an enriched set of regular expression operators. In particular, the Extended Regular Expressions support an operator for alternation, thus allowing a match of one expression or another. It is also important to note that the parenthesis syntax is different from Basic Regular Expressions, and the semantics are subtly different. There are no back-references in Extended Regular Expressions.

The following list summarizes the Extended Regular Expressions:

Expression	Meaning
(*expression*)	Match *expression*; useful for altering precedence
.	(Dot) match any single character
c	Match any nonspecial character *c*
c	Match literal *c*. Normally used to escape ERE special characters.
[*charset*]	Match any element of *charset*
*limited_expression**	Match any number of repetitions of *limited_expression*, including zero
limited_expression+	Match 1 to any number of repetitions of *limited_expression*
limited_expression?	*limited_expression* is optional (match 0 or 1 repetition)
limited_expression\{*M*\}	Match exactly *M* repetitions of *limited_expression*
limited_expression\{,*N*\}	Match zero to *N* repetitions of *limited_expression*
limited_expression\{*M*,*N*\}	Match *M* to *N* repetitions of *limited_expression*
expr0expr1	(Concatenation) match *expr0* then *expr1*
expr0|*expr1*	(Alternation) match *expr0 or expr1* (not both)
^*expression*	Match *expression* only at the beginning of a line

Copyright © 2014, QNX Software Systems Limited

Expression	Meaning
expression$	Match *expression* only at the end of a line

For extended regular expressions, a *limited_expression* is restricted to an expression matching a single character or an expression enclosed in parentheses.

Fixed Regular Expressions

Fixed Regular Expressions consist of a set of strings of characters. They don't permit the operators of extended or basic regular expressions. The algorithm used is extremely efficient for locating one of a set of strings within another string. Thus, if you don't need the various operators of basic or extended regular expressions, the fixed expressions are a better choice.

Examples:

Display lines in `Phone.List` containing telephone numbers:

```
grep '[[:digit:]]\{3\}-[[:digit:]]\{4\}' Phone.List
```

Display all the people logged in who are listed in the `MyFriends` file:

```
who | grep -F -f MyFriends
```

Display all occurrences of the words “steve” and “barney” in the `Phone.List` file:

```
grep -F -e steve -e barney Phone.List
```

Exit status:

0

Lines were found matching the expression provided.

>0

An error occurred or no matching lines were found.

Copyright © 2014, QNX Software Systems Limited

gunzip

Uncompress files (UNIX)

Syntax:

```
gunzip [-cfhLlNnqrtvV] [name...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

See *gzip* (p. 921) for a complete listing.

Description:

The `gunzip` utility uncompresses files that have been compressed with `gzip`. It's equivalent to `gzip -d`. See *gzip* (p. 921) for more details.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

gzip

Compress or expand files (UNIX)

Syntax:

Compress/uncompress files:

```
gzip [-cdfhLlNnqrtVv19] [-S suffix] [name...]
```

Uncompress only:

```
gunzip [-cfhLlNnqrtvV] [name...]
```

Cat compressed file:

```
zcat [-hLV] [name...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-c

Write output on standard output; keep original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d

Decompress.

-f

Force compression or decompression even if the file has multiple links or the corresponding file already exists. If -f isn't given, and when not running in the background, `gzip` prompts to verify whether an existing file should be overwritten.

You need to use the -f option when compressing or expanding files in a RAM (`/dev/shmem`) filesystem.

-h

Display a help message.

Copyright © 2014, QNX Software Systems Limited

-L

Display the `gzip` license.

-l

List compressed file contents.

-N

Save or restore the original name and timestamp.

-n

Don't save or restore the original name and timestamp.

-q

Suppress all warnings.

-r

Travel the directory structure recursively. If any of the file names specified on the command line are directories, `gzip` descends into the directory and compresses all the files it finds there (or decompresses them in the case of `gunzip`).

-S *suffix*

Use the given suffix on compressed files.

-t

Test. Check the compressed file integrity.

-V

Version. Display the version number and compilation options.

-v

Verbose. Display the name and percentage reduction for each file compressed.

-*1* or -9

Regulate the speed of compression, where -1 (the number one) indicates the fastest compression method (less compression) and -9 indicates the slowest compression method (optimal compression).

Description:

The `gzip` utility reduces the size of the named files, using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension `.gz`, while keeping

Copyright © 2014, QNX Software Systems Limited

the same ownership modes, access and modification times. (The extension is -z for VMS, z for MSDOS, OS/2 and Atari.) If no files are specified, the standard input is compressed to the standard output. If the new file name is too long, gzip truncates it and keeps the original file name in the compressed file. The gzip utility attempts to compress only regular files. In particular, it ignores symbolic links.

You can use gzip -d, gunzip, or zcat to restore compressed files to their original form.

These utilities are subject to the GNU Public License (GPL). We've included them for use on development systems.

The gunzip utility takes a list of files on its command line and replaces each file whose name ends with .gz or .GZ or -z and which begins with the correct magic number with an uncompressed file without the original extension. This utility also recognizes the special extensions .tgz and .taz as shorthands for .tar.gz or .tar.GZ.

The gunzip utility can currently decompress files created by gzip, zip, compress or pack. The detection of the input format is automatic. When using the first two formats, gunzip checks a 32-bit CRC. For pack, gunzip checks the uncompressed length. The compress format wasn't designed to allow consistency checks. However gunzip is sometimes able to detect a bad .GZ file. If you get an error when uncompressing a .GZ file, don't assume that the .GZ file is correct just because the standard uncompress doesn't complain. This generally means that the standard uncompress doesn't check its input, and happily generates garbage output.

You can use gzip to uncompress files created by zip only if they have a single member compressed with the “deflation” method. This feature is intended only to help conversion of tar.zip files to the tar.gz format. To extract zip files with several members, obtain and use unzip instead of gunzip. (Note that unzip isn't shipped as part of QNX Neutrino.)

The zcat utility is identical to gunzip -c. (On some systems, zcat may be installed as gzcat to preserve the original link to compress.) The zcat utility uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output. It uncompresses files that have the correct magic number whether they have a .gz suffix or not.

The gzip utility uses the Lempel-Ziv algorithm used in zip and PKZIP. The amount of compression obtained depends on the size of the input and the distribution of common substrings. Typically, text such as source code or English is reduced by 60-70%. Compression is generally much better than that achieved by LZW (as used in compress), Huffman coding (as used in pack), or adaptive Huffman coding (compact).

Copyright © 2014, QNX Software Systems Limited

Compression is always performed, even if the compressed file is slightly larger than the original. The worst case expansion is a few bytes for the `gzip` file header, plus 5 bytes every 32K block, or an expansion ratio of 0.015% for large files. The `gzip` utility preserves the mode, ownership and timestamps of files when compressing or decompressing.

You can concatenate multiple compressed files. In this case, `gunzip` extracts all members at once. For example:

```
gzip -c file1  > foo.gz
gzip -c file2 >> foo.gz
```

Then:

```
gunzip -c foo
```

is equivalent to:

```
cat file1 file2
```

In case of damage to one member of a `.gz` file, other members can still be recovered (if the damaged member is removed). However, you can get better compression by compressing all members at once:

```
cat file1 file2 | gzip > foo.gz
```

compresses better than:

```
gzip -c file1 file2 > foo.gz
```

If you want to recompress concatenated files to get better compression, do:

```
zcat old.gz | gzip > new.gz
```

Environment variables:

GZIP

A set of default options for `gzip`. These options are interpreted first and can be overwritten by explicit command line parameters. For example:

```
export GZIP="-8 -v"
```

Exit status:

2

The operation succeeded but perhaps not 100%; a warning was generated in the process.

1

Copyright © 2014, QNX Software Systems Limited

An error occurred; the operation failed.

0

The operation succeeded.

Contributing author:

GNU

Caveats:

The `.gz` extension is already also used by UNIX `pack`. You can link `gzip` to `pcat` to get transparent decompression for programs expecting `.gz` files to be in pack format.

Copyright © 2014, QNX Software Systems Limited

Chapter 9
H

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`hostapd_cli`	http://netbsd.gw.com/cgi-bin/man-cgi?hostapd_cli+8+NetBSD-4.0

This chapter describes the utilities, etc. whose names start with “H”.

Copyright © 2014, QNX Software Systems Limited

ham

High-availability manager

You must be logged in as `root` to start a high-availability manager.

Syntax:

`ham [`*`options`*`]`

Runs on:

QNX Neutrino

Options:

-?

Display the usage message.

-d

Disable internal verbosity.

-f ***file***

The log file (the default is standard error).

-h

Display the usage message.

-t none | relative | absolute | shortabs

The timestamping method to use. The default is relative.

-V ***level***

Set the level of verbosity.

-v

Be verbose; extra -v options increase the verbosity.

Copyright © 2014, QNX Software Systems Limited

Description:

The ham utility is the high-availability manager, which you can use to monitor and restart critical processes in your system. When a HAM starts, it also starts the Guardian process for itself.

To stop the HAM, you must use either the *ham_stop()* function (see the High Availability Framework *Developer's Guide*) or the *hamctrl* (p. 930) utility. These are the only correct (and the only guaranteed) ways to stop the HAM.

Copyright © 2014, QNX Software Systems Limited

hamctrl

Control a high-availability manager

You must be logged in as `root` to use this utility.

Syntax:

```
hamctrl [options]
```

Runs on:

QNX Neutrino

Options:

-node *node_name*

Control the high-availability manager (HAM) on the specifed node.

-stop

Stop the HAM.

+|-verbose

Increase or decrease the HAM's verbosity.

=verbose

Get the HAM's current verbosity.

Description:

You can use the `hamctrl` utility to control or stop a high-availability manager, *ham* (p. 928).

Copyright © 2014, QNX Software Systems Limited

hd

Display files in decimal, hex, octal, or ASCII (UNIX)

Syntax:

```
hd [-8] [-A format] [-n count] [-s skip]
   [-t format[fmt_string]] [-v] [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-8

Use 8-bit ASCII characters (default 7).

-A *format*

Display the file offset field in the specified format. Valid formats are:

- d — decimal, 9 digits
- n — none (omit this field)
- o — octal, 10 digits
- x — hexadecimal, 7 digits.

-n *count*

Display only *count* bytes of input. You can add a trailing character to specify units of blocks (b), kilobytes (k), or megabytes (m).

-s *skip*

Ignore the first *skip* bytes of data. You can add a trailing character to specify units of blocks (b), kilobytes (k), or megabytes (m).

-t *format*[*fmt_string*]

Use this output/display format; see "*Output formats* (p. 932)", below.

The default format is x1.

-v

Be verbose. If you don't specify the -v option, hd folds multiple identical lines into a single line that contains an asterisk (*).

Copyright © 2014, QNX Software Systems Limited

file

The pathname of an input file. If you don't specify any files, hd reads from standard input. If a *file* is a hyphen (-), hd reads from the standard input at that point in the sequence.

Description:

The hd utility displays data in decimal, hex, octal, or ASCII. The name "hd" (hex dump) is derived from the default output format.

The hd utility processes input in 16-byte units that are formatted into a line. In the default output format:

- the file offset field is displayed in hex, 7 digits
- a space separates the file offset field from the data
- the data is displayed as 16 space-separated bytes in hex
- the same data is also displayed in ASCII, if printable; unprintable data appears as dots.

For example:

```
$ echo "abcdefghijklmnopqrstuvwxyz01234" | hd
0000000: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 abcdefghijklmnop
0000010: 71 72 73 74 75 76 77 78 79 7a 30 31 32 33 34 0a qrstuvwxyz01234.
```

To exclude part of the input, use the -n and -s options. You can specify the arguments to these options in hex (using a 0x prefix) or octal (using a 0 prefix). The default units for these options are bytes, but you can specify different units as follows:

To specify:	Add this suffix:
Blocks (512 bytes)	b
Kilobytes (1024 bytes)	k
Megabytes (1048576 bytes)	m

Output formats

To specify the output format, use the -t option. The *format* argument — which you can specify in decimal, hex, or octal — tells hd which format to use for presenting the output:

a

Named characters. Display printable characters as themselves, and nonprintable characters as a single dot (.).

c

Copyright © 2014, QNX Software Systems Limited

Characters. Display printable characters as themselves; display all other characters as 2-digit hex values, except for the following:

ASCII mnemonic	Value	Representation
NUL	00	\0
<alert>	07	\a
<backspace>	08	\b
<tab>	09	\t
<newline>	0a	\n
<vertical tab>	0b	\v
<formfeed>	0c	\f
<carriage return>	0d	\r

d[1|2|4|C|S|I|L]

Decimal, in objects the size of an `int` by default.

f[4|8|F|D|L]

Floating point, in objects the size of an `float` by default.

o[1|2|4|C|S|I|L]

Octal, in objects the size of an `int` by default.

u[1|2|4|C|S|I|L]

Unsigned decimal, in objects the size of an `int` by default.

x[1|2|4|C|S|I|L]

Hexadecimal, in objects the size of an `int` by default.

The input, processed in 16-byte units formatted into a line, is displayed according to the size you choose:

To display input as:	Choose:
Sixteen 1-byte objects	1
Eight 2-byte objects	2
Four 4-byte values per line	4
Two 8-byte values per line	8
`char`	C

Copyright © 2014, QNX Software Systems Limited

To display input as:	Choose:
`double`	D
`float`	F
`int`	I
`long` or `long double` (depending on the format)	L
`short`	S

Examples:

Display the second to eleventh sectors of the hard disk, `/dev/hd0`:

```
hd -s 1b -n 10b /dev/hd0
```

Exit status:

0

All input files were processed successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

head

Copy the first part of files (POSIX)

Syntax:

```
head [-l] [-number] [-c number] [-n number] [file]...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-number

Deprecated; use -n *number* instead.

-c *number*

The number of bytes to copy. The *number* argument is an unsigned decimal integer.

-l

("el") Measure the quantity of output in lines; this is the default unit of measure.

-n *number*

The number of lines to copy. The *number* argument is an unsigned decimal integer.

file

The pathname of an input file. If you don't specify any files, the standard input is used.

The -c and -l options are QNX Neutrino extensions.

Description:

The head utility copies its input files to the standard output. The utility ends the output for each file at a point designated by the -c or -n option. If you don't specify either of these options, head copies the first ten lines of the file.

Copyright © 2014, QNX Software Systems Limited

If you specify multiple -c, -l, and -n options, the last one takes precedence.

If you specify multiple files, `head` prints an identifying header before the output for each file.

Examples:

Display the first ten lines of all files in the current directory:

```
head *
```

Print the first 16 bytes of `myfile` in hex:

```
head -c 16 myfile | hd
```

(Note that in this case, the same functionality is offered through command-line options to *hd* (p. 931).)

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

hidview

Display information about human-interface devices

- You must be `root` to run this utility.
- You must start `io-hid` *before* you use the `hidview` utility.

Syntax:

```
hidview [-a] [-d devno] [-N name] [-r] [-R]
```

Runs on:

QNX Neutrino

Options:

-a

Display raw data coming from devices.

-d *devno*

Display information about *devno* only.

-N *name*

Name of the HID server to query. The default is `/dev/io-hid/io-hid`.

-R

Display informative report descriptor information.

-r

Display raw report descriptor information.

Description:

The `hidview` utility displays information about human-interface devices (HID).

Examples:

Here's a sample of output from a `hidview` command:

```
HIDD v1.00, v1.00 DDK
```

Copyright © 2014, QNX Software Systems Limited

```
Device Address : 0
Vendor : 0x05c7 (QTRONIX)
Product : 0x2011 (USB Keyboard and Mouse)
Version : r1.00
Usage : Keyboard


Device Address : 1
Vendor : 0x05c7 (QTRONIX)
Product : 0x2011 (USB Keyboard and Mouse)
Version : r1.00
Usage : Mouse
```


Copyright © 2014, QNX Software Systems Limited

hogs

List the processes that are hogging the CPU

Syntax:

```
hogs [options] [pids ...]
```

Runs on:

QNX Neutrino

Options:

-i *iter*

Limit the output to the specified number of iterations (default: no limit).

-m [e][t][p][s]

Specify the type of memory mappings to include in the memory total for each process:

- e — MAP_ELF mappings
- t — MAP_STACK mappings
- s — MAP_SHARED mappings
- p — MAP_PRIVATE mappings (the default)

You can concatenate these types; for example, -msp gives memory for both shared and private mappings.

-n

Show the names of processes (hogs always displays the process IDs).

-p *priority*

Run hogs at the given priority (default: the same as the parent process).

-S [clmlp]

Sort by:

- c — CPU (the default)
- m — memory
- p — process ID

-s *sec*

Copyright © 2014, QNX Software Systems Limited

Sleep this long between updates (default: 3 seconds).

-% *num* [clm]

Show only processes that consume this percentage or more CPU (c, the default) or memory (m). You can use this option to reduce the amount of output.

Description:

The hogs utility displays a list of processes in descending order by percentage of CPU usage (i.e. it shows who's hogging the processor). It loops forever, sleeping between updates.

The format of the output is tabular, and includes:

PID

The ID of the process being reported.

NAME

The name of the process (included only if you specify the -n option).

MSEC

The number of milliseconds for which this process has been running since the last iteration.

PIDS

The amount of time that the process ran in this iteration, as a percentage of the times of all the other processes running.

SYS

The amount of time that the process ran in this iteration, as a percentage of the iteration time.

MEMORY

The amount of memory that the process used. The types of mappings included depend on the -m option.

- The SYS column is incorrect on multicore systems; the numbers in this column will add up to (roughly) the number of processors times 100%. Use the *top* (p. 1966) utility instead.
- The numbers from hogs are approximate. For more precise data, use *tracelogger* (p. 1974) and the System Analysis Toolkit (see the SAT *User's Guide*).

Copyright © 2014, QNX Software Systems Limited

Examples:

Display the processes that are using 10% or more of the CPU. Include the name of the processes:

```
$ hogs -n -% 10c
```

Copyright © 2014, QNX Software Systems Limited

host

DNS lookup utility

Syntax:

```
host [-aCdlnrsTwv] [-c class] [-N ndots] [-R number]
     [-t type] [-W wait] [-m flag] [-4] [-6] {name}
     [server]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?host++NetBSD-5.0* in the NetBSD documentation.

Description:

The `host` utility is for performing DNS lookups. You normally use it to convert names to IP addresses and vice versa. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?host++NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

hostapd

Authenticator for IEEE 802.11 networks

Syntax:

```
hostapd [-BdhKtv] config-file ...
```

Runs on:

QNX Neutrino

Options:

-B

Detach from the controlling terminal and run as a daemon process in the background.

-d

Enable debugging messages. If this option is supplied twice, more verbose messages are displayed.

-h

Show help text.

-K

Include key information in debugging output.

-t

Include timestamps in debugging output.

-v

Display version information on the terminal, and then exit.

config-file

Use the settings in the specified configuration file; the name of the specified wireless interface is contained in this file. See *hostapd.conf* in the NetBSD documentation for a description of the configuration file syntax.

Description:

The hostapd utility is an authenticator for IEEE 802.11 networks. It provides full support for WPA/IEEE 802.11i and can also act as an IEEE 802.1X Authenticator

Copyright © 2014, QNX Software Systems Limited

with a suitable backend Authentication Server (typically FreeRADIUS). The `hostapd` utility implements the authentication protocols that piggyback on top of the normal IEEE 802.11 protocol mechanisms.

To use `hostapd` as an authenticator, the underlying device must support some basic functionality, such as the ability to set security information in the 802.11 management frames. Beware that not all devices have this support.

The `hostapd` utility is designed to be a daemon program that runs in the background and acts as the backend component controlling the wireless connection. It supports separate front-end programs such as the text-based front-end, *`hostapd_cli`*.

You can reload changes to the configuration file by sending a `SIGHUP` signal to the `hostapd` processor, or by using the `hostapd_cli reconfigure` command.

Copyright © 2014, QNX Software Systems Limited

hostname

Set or print the name of the current host system

Syntax:

```
hostname [-s] [name_of_host]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-s

Trim off any domain information from the printed name.

name_of_host

The name to be given to the current host system.

Description:

When the system starts, it initializes the hostname. As superuser, you can use the `hostname` utility to change the hostname.

Without a *name_of_host* argument, `hostname` prints the name of the current host.

This utility sets or gets the value of the `_CS_HOSTNAME` configuration string, not that of the ***HOSTNAME*** environment variable.

Copyright © 2014, QNX Software Systems Limited

/etc/hosts

Hostname database (UNIX)

Name:

`/etc/hosts`

Description:

The `/etc/hosts` file contains information regarding the known hosts on the network. For each host, a single line should be present with the following information:

```
internet_address  official_host_name  aliases
```

Items are separated by any number of spaces or tabs, or both; however, spaces or tabs aren't allowed before the IP address. A # indicates the beginning of a comment — any characters after a #, up to the end of the line, aren't interpreted by routines that search the file.

If you're using a name server, this file provides a backup when the server isn't running. In this case, you should include only a few addresses in this file. These include addresses for the local interfaces that *ifconfig* (p. 957) needs at boot time and a few machines on the local network.

Network addresses are specified in the conventional "." (dot) notation using the *inet_addr()* routine from the internet address manipulation functions. Hostnames may contain any printable character other than a field delimiter, newline, or comment character.

For more information, see *DNS and BIND* by Paul Albitz and Cricket Liu, O'Reilly & Associates (ISBN 1-56592-010-4).

Copyright © 2014, QNX Software Systems Limited

/etc/hosts.equiv

System-wide list of trusted remote hosts

Name:

/etc/hosts.equiv

Description:

The /etc/hosts.equiv and *~/.rhosts* (p. 1666) files provide the "remote authentication" database for the *lpd* (p. 1126), *rcp* (p. 1658), *rlogin* (p. 1669), and *rsh* (p. 1703) commands and the *rcmd()* function. These files bypass the standard password-based user authentication mechanism. They specify remote hosts and users that are considered *trusted* (i.e. are allowed to access the local system without supplying a password):

- on a system-wide basis (/etc/hosts.equiv)
- by an individual user (~/.rhosts).

The file permissions for the ~/.rhosts file must be as follows or its contents will be ignored:

- it must be owned by root or the user
- it cannot be writable by anyone other than the owner (e.g. rw-r--r--)

The *ruserok()* function sets the effective userid to that of the remote user, but doesn't change the effective group ID. The user must have search permissions for the directories contained in the pathname of an .rhosts file (i.e. if the file resides in /home/user/.rhosts, the user must have search permissions for /home/user/).

The library routine *ruserok()* (see also *rcmd()*) performs the remote authentication. It determines whether a particular remote user from a particular remote host is allowed to access the local system as a (possibly different) particular local user:

- For non-root users, this routine checks /etc/hosts.equiv, and then the .rhosts file in the home directory of the local user attempting access.
- For root, access is handled as a special case to help maintain system security; only root's .rhosts file is checked.

The *rlogind* (p. 1671) daemon doesn't allow root to log in without a password. When rsh is specified without command options, rlogind (not rshd) is invoked on the remote side.

Copyright © 2014, QNX Software Systems Limited

If the remote authentication fails, `lpd`, `rcp` and `rsh` fail, but `rlogin` falls back to the standard password-based login procedure.

Both files are formatted as a list of one-line entries of the form:

```
hostname [username]
```

where *hostname* must be the fully qualified domain name (FQDN) of the host, not one of its aliases.

The entries in these files are either *positive*, to explicitly allow access without a password, or *negative*, to deny it. Authentication succeeds as soon as a matching positive entry is found, but fails when a matching negative entry is found, or if no matching entries are found in either file. Therefore, the order of entries is important: if the files contain both matching positive and negative entries, the entry that appears first prevails.

Positive entries

Positive entries take these forms:

hostname

All users from the named host are trusted and may access the system with the same user name as they have on the remote system. You can use this form in both `/etc/hosts.equiv` and individual users' `.rhosts` files.

hostname username

The meaning of this form depends on which file it's in:

- `.rhosts` file in a local user's home directory — the named user from the named host can access the system as that local user.
- `/etc/hosts.equiv` — the named remote user can access the system as *any* local user.

You can use the special character "+" as a wild card in place of either *hostname* or *username* to match any host or user:

+

Any user from any remote host can access the system, with the same username.

+ *username*

The named user from any remote host can access the system.

***hostname* +**

Any user from the named host can access the system as the local user.

Copyright © 2014, QNX Software Systems Limited

Negative entries

Negative entries have a "-" character preceding either the *hostname* or *username* field. For example:

hostname -username

Deny access to the named user if they attempt to access your system from the named host without providing a password.

Caveats:

Use extreme caution in `/etc/hosts.equiv` with positive entries that include a username field (either an individual named user, a netgroup, or "+" sign). Because `/etc/hosts.equiv` applies system-wide, these entries allow one or a group of remote users to access the system as any local user without providing a password. This can be a security hole.

The file permissions for the `~/.rhosts` file must be as follows or its contents will be ignored:

- it must be owned by `root` or the user
- it cannot be writable by anyone other than the owner (e.g. `rw-r--r--`)

Copyright © 2014, QNX Software Systems Limited

Chapter 10
I

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`info`	No replacement
`io-net`	*`io-pkt*`* (p. 1007)
`ipf`, `ipfs`, `ipfstat`, `ipmon`, `ipnat`	*`pf`* (p. 1461), *`pf.conf`* (p. 1476), and *`pfctl`* (p. 1513)

This chapter describes the utilities, etc. whose names start with “I”.

Copyright © 2014, QNX Software Systems Limited

id

Return user and group IDs (POSIX)

Syntax:

```
id [username]
id -G [-n] [username]
id -g [-nr] [username]
id -u [-nr] [username]
```

Runs on:

QNX Neutrino

Options:

-G

Write the real (and effective, if different) group ID only.

-g

Write the effective group ID only.

-u

Write the effective user ID only.

-n

In combination with -G, -g, or -u, write the ID as its name instead of as an unsigned integer.

-r

In combination with -g or -u, write the real ID instead of the effective ID.

Description:

The `id` utility writes the current real and/or effective user ID and group ID. When no options are specified, output is as follows:

```
uid=nnn(username) gid=nnn(groupname)
```


Copyright © 2014, QNX Software Systems Limited

If the effective user ID is different from the real user ID, the effective user ID also appears:

```
... euid=nnn(effective_username)
```

Likewise, if the effective group ID is different from the real group ID, the effective group ID appears as well:

```
... egid=nnn(effective_groupname)
```

If no entry exists for an ID in the `/etc/passwd` or `/etc/group` files, the output lacks the (*name*) after the numerical value. No error is generated.

If a *username* is supplied as an operand, the effective *uid* and *gid* don't appear since there's no process associated with it — the information is simply looked up in the `/etc/passwd` and `/etc/group` files.

When options are specified, the requested data is written as an unsigned integer, unless option -n is specified, in which case the data is written as the corresponding user or group name. With option -G, the `id` utility might produce two lines of output (one value per line) if the real and effective IDs differ. In all other cases, `id` always produces one line of output.

Examples:

Write information on current IDs (real and effective):

```
$ id
uid=109(eric) gid=120(techies)
```

Write the effective group ID as a number:

```
$ id -g
120
```

Write the effective group ID as a name:

```
$ id -gn
techies
```

Files:

`/etc/passwd`

Password file; defines user IDs, home directories, etc.

`/etc/group`

Group file; defines the valid group IDs for the system, also lists user IDs who may change to each group.

Exit status:

Copyright © 2014, QNX Software Systems Limited

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

if_up

Ensure a TCP/IP interface is available

If you aren't `root`, specify the full path: `/usr/sbin/if_up`.

Syntax:

`if_up [-alp] [-r retries] [-s seconds] interface...`

Runs on:

QNX Neutrino

Options:

-a

Wait until all specified interfaces are configured. The default is to wait until a single interface is configured.

-l

("el") Wait for the network link to be marked as being up. This causes `if_up` to wait until the physical network link is up. This option may not be supported by some drivers.

-p

Wait only until the specified interfaces are present. The default is to wait until the interfaces are both present and configured. This is useful if you intend to configure the interface manually, after it is present (e.g. using the *`ifconfig`* (p. 957) utility).

-r *retries*

The number of times to walk the interface list. The default is 5.

-s *seconds*

Wait this number of seconds before rewalking the interface list. The default is 1.

interface

The name of the interface to wait on (e.g. `en0`, `en1`, ...).

Copyright © 2014, QNX Software Systems Limited

Description:

You can use this convenience utility while booting to ensure a TCP/IP interface is available to those utilities that require one. It's primarily intended for use with the *dhcp.client* (p. 544). When `if_up` is run in the foreground in a startup script, the script doesn't continue until the specified interfaces are marked as "UP" and have been assigned an IP address, or the utility has timed out.

Examples:

See *dhcp.client* (p. 544).

Files:

`libsocket.so`

The `if_up` utility requires the `libsocket.so` shared library.

`/usr/sbin/if_up`

The `if_up` utility is located in the `/usr/sbin/` directory, which isn't included in the default ***PATH*** of non-root users. If you aren't `root`, specify the full path.

Exit status:

0

Success.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

ifconfig

Configure network interface parameters

Syntax:

```
ifconfig interface address_family [address [dest_address]]
         [parameters]
ifconfig [-hLmvz] interface [protocol_family]
ifconfig -a [-bdhLmsuvz] [protocol_family]
ifconfig -l [-b] [-d] [-u] [-s]
ifconfig -s interface
ifconfig -C
```

Runs on:

QNX Neutrino

Options:

-a

Display information about all of the interfaces in the system. You can use the -d, -u, -b, and -s options with this option to limit this display.

-b

List only the broadcast interfaces.

-C

List all of the interface cloners available on the system, with no additional information. This option is mutually exclusive with all other options and commands.

-d

List only the interfaces that are down.

-h

If you use this option in conjunction with -v, `ifconfig` prints the byte statistics are in human-readable format.

-L

Display the address lifetime for IPv6 addresses, as a time offset string.

-l

Copyright © 2014, QNX Software Systems Limited

List all available interfaces on the system, with no additional information. This option is mutually exclusive with all other options and commands, except for -b, -d, -s, -u.

-m

Display all of the supported media for all of the interfaces in the system (used in conjunction with -a).

If you specify the -m option before an interface name, `ifconfig` displays all of the supported media for the specified interface.

-s

If you specify the -a option, the -s option makes `ifconfig` list only the interfaces that are connected.

If you specify the -s option for a specific interface, `ifconfig` queries the interface for its media status. If the interface supports reporting media status, and it reports that it doesn't appear to be connected to a network, `ifconfig` exits with status of 1 (false); otherwise, it exits with zero (true). Not all interface drivers support media status reporting.

-u

List only the interfaces which are up.

-v

Print statistics on packets sent and received on the given interface. You can use the -h in conjunction with -v to display the byte statistics in human-readable format.

-z

Similar to the -v flag, except that it zeros the interface's input and output statistics after printing them.

interface

The name of the interface to configure. This is a string of the form *name unit* (e.g. `en1`)

address

Either a hostname present in the *`/etc/hosts`* (p. 946) database, or a DARPA-Internet address expressed in the standard Internet “dot notation.”

For the Xerox Network Systems family, addresses are in the form *net:a.b.c.d.e.f*, where *net* is the assigned network number (in decimal), and each of the six bytes of the host number, *a* through *f*, are specified in

Copyright © 2014, QNX Software Systems Limited

hexadecimal. The host number may be omitted on Ethernet interfaces, which use the hardware physical address, and on interfaces other than the first.

For the ISO family, addresses are specified as a long hexadecimal string, as in the Xerox family. However, two consecutive dots imply a zero byte, and the dots are optional, if you wish to (carefully) count out long strings of digits in network byte order.

address_family

The address family that affects the interpretation of the remaining parameters. Specifying an address family is recommended since an interface can receive transmissions in differing protocols with different naming schemes. Address or protocol families currently supported are `inet`, `inet6`, `atalk`, `iso`, and `ns`.

dest_address

Address of the correspondent on the other end of a point-to-point link (for ppp*x* interfaces only).

parameters

See the "*Parameters* (p. 959)" section below.

protocol_family

Report only the details specific to this protocol family.

Description:

The `ifconfig` utility is used to assign an address and/or configure parameters for a network interface. This utility must be run at boot time to define the network address of each interface present on a machine; it may also be run later on to redefine an interface's address or to configure other interface parameters.

When no optional parameters are specified, `ifconfig` displays the current configuration for a network interface. If you specify a protocol family, `ifconfig` reports only the details specific to that protocol family.

Only the superuser can modify the configuration of a network interface.

Parameters

You may set the following parameters with the `ifconfig` utility, if the driver supports them. The output from `ifconfig` for an interface lists the supported parameters.

`alias`

Copyright © 2014, QNX Software Systems Limited

Establish an additional network address for this interface. This is useful when someone changes network address of an interface, or when someone wishes to accept packets addressed to the old interface.

-alias

Remove the additional network address for this interface.

anycast

(`inet6` only) Set the IPv6 anycast address bit.

-anycast

(`inet6` only) Clear the IPv6 anycast address bit.

apbridge

(IEEE 802.11 devices only) When operating as an access point, pass packets between wireless clients directly (the default).

-apbridge

(IEEE 802.11 devices only) When operating as an access point, pass packets through the system so that they can be forwarded using some other mechanism. Disabling the internal bridging is useful when traffic is to be processed with packet filtering.

arp

Enable the use of the Address Resolution Protocol in mapping between network-level addresses and link-level addresses (default). This is implemented to do mapping between DARPA Internet addresses and Ethernet addresses.

-arp

Disable the use of the Address Resolution Protocol.

broadcast ***mask***

(`inet` only) Use this address to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

bssid ***bssid***

(IEEE 802.11 devices only) Set the desired BSSID for IEEE 802.11-based wireless network interfaces.

-bssid

Copyright © 2014, QNX Software Systems Limited

(IEEE 802.11 devices only) Unset the desired BSSID for IEEE 802.11-based wireless network interfaces. The interface will automatically select a BSSID in this mode, which is the default.

chan ***chan***

(IEEE 802.11 devices only) Select the channel (radio frequency) to use for IEEE 802.11-based wireless network interfaces.

-chan

(IEEE 802.11 devices only) Unset the desired channel to to be used for IEEE 802.11-based wireless network interfaces. It doesn't affect the channel to be created for IBSS or hostap mode.

chanlist ***channels***

Set the channels to use when scanning for access points, neighbors in an IBSS network, or looking for unoccupied channels when operating as an access point.

Specify the set of channels as a comma-seperated list, with each element in the list representing either a single channel number of a range of the form *a*-*b*. Channel numbers must be in the range 1 through 255 and be permissible according to the operating characteristics of the device.

create

Create the specified network pseudo-device.

debug

Enable driver-dependent debugging code; usually, this turns on extra console error logging.

-debug

Disable driver-dependent debugging code.

delete

Remove a specified network address. You should use this parameter if you have incorrectly specified an alias, or you will no longer use an alias. In the event that you have incorrectly set an NS address which has the side effect of specifying the host portion, you must respecify the host portion while removing all NS addresses. Note that this parameter doesn't work for IPv6 addresses. If you need to delete IPv6 addresses, use `-alias` with an explicit IPv6 address.

deletetunnel

Copyright © 2014, QNX Software Systems Limited

Unconfigure the physical source and destination address for IP tunnel interfaces previously configured with `tunnel`.

`deprecated`

(`inet6` only) Set the IPv6 deprecated address bit.

`-deprecated`

(`inet6` only) Clear the IPv6 deprecated address bit.

`destroy`

Destroy the specified network pseudo-device.

`dest_address`

Specify the address of the correspondent on the other end of a point-to-point link.

`down`

Mark an interface as being down. When an interface is marked down, the system doesn't attempt to transmit messages through that interface. If possible, the interface is reset to disable reception as well. This action doesn't automatically disable routes using the interface.

`eui64`

(`inet6` only) Fill the interface index (the lowermost 64 bits of an IPv6 address) automatically.

`hidessid`

(IEEE 802.11 devices only) When operating as an access point, don't broadcast the SSID in beacon frames or respond to probe request frames unless they're directed to the access point (i.e. they include its SSID). By default, the SSID is included in beacon frames, and undirected probe request frames are answered.

`-hidessid`

(IEEE 802.11 devices only) When operating as an access point, broadcast the SSID in beacon frames and answer and respond to undirected probe request frames (default).

`instance` ***minst***

Set the media instance to *minst*. This is useful for devices that have multiple physical layer interfaces (PHYs). Setting the instance on such devices may not be strictly required by the network interface driver because the driver

Copyright © 2014, QNX Software Systems Limited

may take care of this automatically; see the driver's documentation for more information.

ip4csum, ip4csum-rx, ip4csum-tx

Enable hardware-assisted IPv4 header checksums, if they're supported. You can restrict this action to either the Rx or Tx direction, if the hardware permits it.

-ip4csum, -ip4csum-rx, -ip4csum-tx

Disable hardware-assisted IPv4 header checksums.

ipdst

Specify an Internet host that's willing to receive IP packets encapsulating NS packets bound for a remote network. An apparent point-to-point link is constructed, and the address specified is taken as the NS address and network of the destination. IP encapsulation of CLNP packets is done differently.

link *mac* **[active|delete]**

Dynamically modify the specified interface's MAC addresses. If you don't specify `active` or `delete`, `ifconfig` adds the given MAC address. The `active` command activates the address, and the `delete` command removes it.

link[0-2]

Enable special processing of the link level of the interface. These three options are interface-specific in actual effect, but you use them in general to select special modes of operation. An example of this is to select the connector type for some ethernet cards. For more information, see the documentation for the specific driver.

-link[0-2]

Disable special processing at the link level with the specified interface.

media *type*

Set the media type of the interface to type. Some interfaces support the mutually exclusive use of one of several different physical media connectors. For example, a 10 Mb/s Ethernet interface might support the use of either AUI or twisted-pair connectors. Setting the media type to `10base5` or `AUI` would change the currently active connector to the AUI port. Setting it to `10baseT` or `UTP` would activate twisted pair. Refer to the interfaces' driver-specific documentation for a complete list of the available types.

Copyright © 2014, QNX Software Systems Limited

mediaopt *opts*

Set the specified media options on the interface. The *opts* argument is a comma-delimited list of options to apply to the interface. For information about the available options, see the documentation for each driver.

-mediaopt *opts*

Disable the specified media options on the interface.

metric *n*

Set the routing metric of the interface to *n*. The default is 0. The routing metric is used by the routing protocol (*routed* (p. 1688)). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

mode *mode*

If the driver supports the media selection system, set the specified operating mode on the interface to *mode*. For IEEE 802.11 wireless interfaces that support multiple operating modes, this directive is used to select between 802.11a (`11a`), 802.11b (`11b`), and 802.11g (`11g`) operating modes.

mtu *n*

Set the maximum transmission unit of the interface to *n*. Most interfaces don't support this parameter.

netmask *mask*

(inet, inet6, and ISO) Specify how much of the address to reserve for subdividing networks into subnetworks. The *mask* includes the network part of the local address and the subnet part, which is taken from the host field of the address. You can specify the *mask* as a single hexadecimal number with a leading `0x`, with a dot-notation Internet address, or with a pseudo-network name listed in the network table, `networks`.

The mask contains ones for the bit positions in the 32-bit address that are to be used for the network and subnet parts, and zeroes for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

For INET and INET6 addresses, you can also give the netmask with slash-notation after the address (e.g. 192.168.17.3/24).

nsellength *n*

Copyright © 2014, QNX Software Systems Limited

(ISO only) This specifies a trailing number of bytes for a received NSAP used for local identification, the remaining leading part of which is taken to be the NET (Network Entity Title). The default value is 1, which is conformant to US GOSIP. When an ISO address is set in an `ifconfig` command, it's really the NSAP that's being specified. For example, in US GOSIP, 20 hex digits should be specified in the ISO NSAP to be assigned to the interface. There is some evidence that a number different from 1 may be useful for AFI 37 type addresses.

`nwid` ***id***

A synonym for `ssid`.

`nwkey` ***key***

(IEEE 802.11 devices only) Enable WEP encryption for IEEE 802.11-based wireless network interfaces with the key. The key can either be a string, a series of hexadecimal digits preceded by `0x`, or a set of keys in the form *n:k1,k2,k3,k4*, where *n* specifies which of keys will be used for all transmitted packets, and four keys, *k1* through *k4*, are configured as WEP keys. Note that the order must be matched within same network if you use multiple keys. For IEEE 802.11 wireless network, the length of each key is restricted to 40 bits, i.e. a 5-character string or 10 hexadecimal digits, while the WaveLAN/IEEE Gold cards accept the 104 bits (13 characters) key.

`nwkey` ***persist***

(IEEE 802.11 devices only) Enable WEP encryption for IEEE 802.11-based wireless network interfaces with the persistent key written in the network card.

`nwkey` ***persist:key***

(IEEE 802.11 devices only) Write the key to the persistent memory of the network card, and enable WEP encryption for IEEE 802.11-based wireless network interfaces with the key.

`-nwkey`

(IEEE 802.11 devices only) Disable WEP encryption for IEEE 802.11-based wireless network interfaces.

`pltime` ***n***

(`inet6` only) Set the preferred lifetime for the address, in seconds.

`powersave`

(IEEE 802.11 devices only) Enable 802.11 power-saving mode.

Copyright © 2014, QNX Software Systems Limited

-powersave

(IEEE 802.11 devices only) Disable 802.11 power-saving mode.

powersavesleep *duration*

(IEEE 802.11 devices only) Set the receiver sleep duration, in milliseconds, for the 802.11 power-saving mode.

prefixlen *n*

(`inet` and `inet6` only) Similar to `netmask`, but you can specify the length of the prefix.

ssid *id*

(IEEE 802.11 devices only) Configure the Service Set Identifier (also known as the network name) for IEEE 802.11-based wireless network interfaces. The *id* can either be any text string up to 32 characters in length, or a series of up to 64 hexadecimal digits preceded by `0x`. Setting *id* to the empty string allows the interface to connect to any available access point.

tcp4csum, tcp4csum-rx, tcp4csum-tx

Enable hardware-assisted TCP/IPv4 checksums, if they're supported. You can restrict this action to either the Rx or Tx direction, if the hardware permits it.

-tcp4csum, -tcp4csum-rx, -tcp4csum-tx

Disable hardware-assisted TCP/IPv4 checksums.

tcp6csum, tcp6csum-rx, tcp6csum-tx

Enable hardware-assisted TCP/IPv6 checksums, if they're supported. You can restrict this action to either the Rx or Tx direction, if the hardware permits it.

-tcp6csum, -tcp6csum-rx, -tcp6csum-tx

Disable hardware-assisted TCP/IPv6 checksums.

tentative

(`inet6` only) Set the IPv6 tentative address bit.

-tentative

(`inet6` only) Clear the IPv6 tentative address bit.

tso4

Copyright © 2014, QNX Software Systems Limited

Enable hardware-assisted TCP/IPv4 segmentation on interfaces that support it.

-tso4

Disable hardware-assisted TCP/IPv4 segmentation on interfaces that support it.

tunnel ***src_addr*[,*src_port*] *dest_addr*[,*dest_port*]**

(IP tunnel devices only) Configure the physical source and destination address for IP tunnel interfaces, including *gif*. The arguments, *src_addr* and *dest_addr* are interpreted as the outer source and destination for the encapsulating IPv4/IPv6 header.

On a *gre* interface in UDP mode, the arguments *src_port* and *dest_port* are interpreted as the outer source and destination port for the encapsulating UDP header.

udp4csum, udp4csum-rx, udp4csum-tx

Enable hardware-assisted UDP4 checksums, if they're supported. You can restrict this action to either the Rx or Tx direction, if the hardware permits it.

-udp4csum, -udp4csum-rx, -udp4csum-tx

Disable hardware-assisted UDP4 checksums.

udp6csum, udp6csum-rx, udp6csum-tx

Enable hardware-assisted UDP6 checksums, if they're supported. You can restrict this action to either the Rx or Tx direction, if the hardware permits it.

-udp6csum, -udp6csum-rx, -udp6csum-tx

Disable hardware-assisted UDP6 checksums.

up

Mark an interface as being up. You can use this to enable an interface after an `ifconfig down` command. By default, an interface is marked as "up" the first time `ifconfig` is run to assign the interface an address. If the interface was reset when previously marked down, the hardware is reinitialized.

vlan ***tag***

If the interface is a *vlan* pseudo-interface, set the VLAN tag to *tag*. This is a 12-bit number which is used to create an 802.1Q VLAN header for packets

Copyright © 2014, QNX Software Systems Limited

sent from the `vlan` interface. Note that you must set `vlan` and `vlanif` at the same time.

`vlanif` *iface*

If the interface is a `vlan` pseudo-interface, associate the physical interface, *iface*, with it. Packets transmitted through the `vlan` interface will be diverted to the specified physical interface iface with 802.1Q VLAN encapsulation. Packets with 802.1Q encapsulation received by the physical interface with the correct VLAN tag will be diverted to the associated `vlan` pseudo-interface.

The VLAN interface is assigned a copy of the physical interface's flags and Ethernet address. If the `vlan` interface already has a physical interface associated with it, this command will fail. To change the association to another physical interface, the existing association must be cleared first.

Note that you must set `vlan` and `vlanif` at the same time.

`vltime` *n*

(`inet6` only) Set the valid lifetime for the address.

The following parameters are specific to IEEE 802.11 wireless interfaces:

`list active`

Display the list of channels available for use, taking into account any restrictions set with the `chanlist` directive. See the description of `list chan` for more information.

`list caps`

Display the adaptor's capabilities, including the operating modes supported.

`list chan`

Display the list of channels available for use. Channels are shown with their IEEE channel number, equivalent frequency, and usage modes. Channels identified as “11g” are also usable in “11b” mode. Channels identified as “11a Turbo” may be used only for Atheros' Static Turbo mode (specified with mediaopt turbo). Channels marked with a * have a regulatory constraint that they be passively scanned. This means a station isn't permitted to transmit on the channel until it identifies the channel as being used for 802.11 communication, typically by hearing a beacon frame from an access point operating on the channel. You can also use `list freq` to request this information.

`list mac`

Copyright © 2014, QNX Software Systems Limited

Display the current MAC Access Control List state. Each address is prefixed with a character that indicates the current policy applied to it:

This character:	Indicates:
+	The address is allowed access.
-	The address is denied access.
*	The address is present, but the current policy is open (so the ACL isn't consulted).

list scan

Display the access points and/or ad hoc neighbors located in the vicinity. You can use the -v option to display long SSIDs. This information may be updated automatically by the adaptor and/or with a scan request. You can also use `list ap` to request this information.

list sta

When operating as an access point, display the stations that are currently associated. When operating in ad hoc mode, display stations identified as neighbors in the IBSS. Capabilities advertised by the stations are described under the scan request. Depending on the capabilities of the stations the following flags can be included in the output:

A

Authorized. Indicates that the station is permitted to send/receive data frames.

E

Extended Rate Phy (ERP). Indicates that the station is operating in an 802.11g network using extended transmit rates.

P

Power Save. Indicates that the station is operating in power-saving mode.

scan

Initiate a scan of neighboring stations, wait for it to complete, and display all stations found. Only the superuser can initiate a scan. Depending on the capabilities of the APs, the following flags can be included in the output:

E

Copyright © 2014, QNX Software Systems Limited

Extended Service Set (ESS). Indicates that the station is part of an infrastructure network (in contrast to an IBSS/ad hoc network).

I

IBSS/ad hoc network. Indicates that the station is part of an ad hoc network (in contrast to an ESS network).

P

Privacy. Data confidentiality is required for all data frames exchanged within the BSS. This means that this BSS requires the station to use cryptographic means such as WEP, TKIP, or AES-CCMP to encrypt and decrypt data frames being exchanged with others.

S

Short Preamble. Indicates that the network is using short preambles (defined in 802.11b High Rate/DSSS PHY). A short preamble uses a 56-bit sync field, in contrast to a 128-bit field used in long preamble mode.

s

The network is using a short slot time.

You can use the `list scan` request to show recent scan results without initiating a new scan. You can use the -v option to prevent the shortening of long SSIDs.

Diagnostics

Depending on the error, the utility may display messages indicating:

- the specified interface doesn't exist
- the requested address is unknown
- the user isn't privileged and tried to alter an interface's configuration

Examples:

Enable hardware-assisted TCP/IPv6 checksums on an interface:

```
ifconfig wm0 tcp6csum
```


Copyright © 2014, QNX Software Systems Limited

ifwatchd

Watch for addresses added to or deleted from interfaces and call up/down-scripts for them

Syntax:

```
ifwatchd [-hiqv] [-A arrival-script] [-c carrier-script]
         [-D departure-script] [-d down-script]
         [-n no-carrier-script] [-u up-script] ifname(s)
```

Runs on:

QNX Neutrino

Options:

-A ***arrival-script***

Specify the command to invoke when new interfaces (such as PCMCIA cards) arrive.

-c ***carrier-script***

Specify the command to invoke when the carrier status changes from no carrier to carrier.

-D ***departure-script***

Specify the command to invoke when an interface departs (for example a PCMCIA card is removed.)

-d ***down-script***

Specify the command to invoke on “interface down” events (or on the deletion of an address from an interface).

-h

Show the synopsis.

-i

Inhibit a call to the up-script on startup for all watched interfaces already marked up. If you don't specify this option, `ifwatchd` checks all watched interfaces on startup whether they're already marked up and, if they are, calls the up-script with appropriate parameters.

Copyright © 2014, QNX Software Systems Limited

Since `ifwatchd` typically is started late in the system boot sequence, some of the monitored interfaces may already have come up when it finally starts, but their up-scripts haven't been called. By default, `ifwatchd` calls them on startup to account for this (and make the scripts easier.)

-n ***no-carrier-script***

Specify the command to invoke when the carrier status transitions from carrier to no carrier.

-q

Be quiet and don't log non-error messages to *slogger* (p. 1807).

-u ***up-script***

Specify the command to invoke on "interface up" events (or on the addition of an address to an interface).

-v

Run in verbose debug mode and don't detach from the controlling terminal. Output verbose progress messages, and flag those errors that are ignored during normal operation. Don't use this option in `/etc/rc.conf`.

ifname(s)

The name of the interface to watch. You can specify multiple interfaces. Events for other interfaces are ignored.

Description:

The `ifwatchd` utility is used to monitor dynamic interfaces (for example PPP interfaces) for address changes, and to monitor static interfaces for carrier changes. Sometimes these interfaces are accompanied by a daemon program, which can take care of running any necessary scripts (such as `pppd` or `isdnd`), but sometimes the interfaces run completely autonomously (such as `pppoe`).

The `ifwatchd` utility provides a generic way to watch these types of changes. It works by monitoring the routing socket and interpreting `RTM_NEWADDR` (address added), `RTM_DELADDR` (address deleted) and `RTM_IFINFO` (carrier detect or loss of carrier) messages. It doesn't need special privileges to do this. The scripts called for up or down events are run with the same user ID as for `ifwatchd`.

Examples:

```
ifwatchd -u /etc/ppp/ip-up -d /etc/ppp/ip-down pppoe0
```


Copyright © 2014, QNX Software Systems Limited

If your pppoe0 interface is your main connection to the Internet, the typical use of the up/down scripts is to add and remove a default route. This is an example of an up script that does this:

```
#! /bin/sh
/sbin/route add default $5
```

As described below, the fifth command line parameter contains the peer address of the pppoe link. The corresponding ip-down script is:

```
#! /bin/sh
/sbin/route delete default $5
```

This isn't a good idea if you have pppoe0 configured to connect only on demand (via the link1 flag), but works well for all permanent connected cases. Use:

```
! /sbin/route add default -iface 0.0.0.1
```

in your /etc/ifconfig.pppoe0 file in the on-demand case.

Copyright © 2014, QNX Software Systems Limited

indent

Format C source

Syntax:

```
indent [input-file [output-file]] [options...]
```

Runs on:

QNX Neutrino

Options:

The first option of a pair is described; the second option disables the first. The second option (disabled) is default.

-bad, -nbad

Force a blank line after each block of declarations.

-bap, -nbap

Force a blank line after procedure bodies.

-bbb, -nbbb

Force a blank line before every block comment.

-nbc, -bc

Don't force a newline after each comma in a declaration.

-bl, -br

Move the first brace ({) of a compound statement to the next line.

-c*n*

Specify the column in which comments on code start.

-cd*n*

Specify the column in which comments on declarations start.

-ncdb, -cdb

Don't place comment delimiters alone on blank lines.

-nce, -ce

Don't place `else` statements adjacent to the preceding closing brace (}).

Copyright © 2014, QNX Software Systems Limited

-ci*n*

Set the line overflow indentation to be *n*.

-cli*n*

Indent `case` labels *n* tab stops past the `switch`.

-d1

Indent code comments by themselves one tab stop.

-di*n*

Specify the column indentation of variable declarations.

-ndj, -dj

Left-justify declarations.

-ei, -nei

Indent an `if` following an `else` the same as the preceding `if`.

-nfc1, -fc1

Don't format comments that start in column 1.

-i*n*

Specify the number of spaces for one indentation level. The default is 4.

-nip, -ip

Disable the indentation of parameter declarations.

-l*n*

Specify the maximum length of an output line. The default is 75.

-nlp, -lp

Don't line up parameters that overflowed onto the next line.

-npro

Ignore the profile files `./.indent.pro` and `~/.indent.pro`.

-pcs, -npcs

Place a space between procedure names and the following parenthesis (`(`).

-npsl, -psl

Don't place column 1 procedure types on the previous line.

Copyright © 2014, QNX Software Systems Limited

-nsc, -sc

Disable the placement of asterisks (*) at the left edge of comments.

-sob, -nsob

Remove optional blank lines.

-st

Take input from *stdin*, and send output to *stdout*.

-T*typename*

Add *typename* to the list of type keywords.

-troff

Format for processing by `troff`. If you use this option, the output is sent to *stdout* by default.

-v, -nv

Be verbose; provide statistics and line-splitting information.

Description:

The `indent` utility formats C source code.

If you specify only *input-file*, it's modified.

Files:

`./.indent.pro`, `~/.indent.pro`

Profile files.

Copyright © 2014, QNX Software Systems Limited

inetd

Internet super-server (UNIX)

You must be `root` to start this daemon.

Syntax:

`inetd [-Dd] [`*`configuration_file`*`]`

Runs on:

QNX Neutrino

Options:

-D

Force `inetd` to daemonize by calling *procmgr_daemon()* instead of calling *daemon()*.

You need to specify the -D option to `inetd` if you're running it under the control of the High Availability Manager. The HAM can see death messages only from tasks that are running in session 1, and the call to *daemon()* doesn't put the caller into that session. For more information about HAM, see the High Availability Framework *Developer's Guide*.

-d

Turn on debugging.

Description:

The `inetd` daemon listens for connections on certain well-known ports. When it finds a connection on one of its sockets, the daemon decides what service the socket corresponds to and invokes a program to service the request. After that program is finished, `inetd` continues to listen on the socket (except in some cases, described below). Essentially, `inetd` lets you run one daemon to invoke several others, reducing load on the system.

When it starts, `inetd` reads its configuration information from a configuration file; by default, this is *`/etc/inetd.conf`* (p. 979).

Copyright © 2014, QNX Software Systems Limited

The default version of the `/etc/inetd.conf` file contains commented-out descriptions of the services; uncomment the ones that you want to use.

If any errors occur, `inetd` sends messages to *`slogger`* (p. 1807); use *`sloginfo`* (p. 1818) to view the system log.

Internal services

The `inetd` daemon provides several "trivial" services internally by using routines within itself. These services are:

`echo`

Echo the data received.

`discard`

Discard the data received.

`chargen`

Generate characters.

`daytime`

Human-readable time.

`time`

Machine-readable time, in the form of the number of seconds since midnight, January 1, 1900.

All of these services are UDP- or TCP-based.

Effects of `SIGHUP`

When it receives `SIGHUP`, `inetd` rereads its configuration file, which may cause services to be added, deleted, or modified.

Files:

The `inetd` daemon requires the `libsocket.so` shared library.

If you use RPC-based services, the `librpc.so` shared library must exist.

Copyright © 2014, QNX Software Systems Limited

/etc/inetd.conf

Super-server configuration file (UNIX)

Name:

`/etc/inetd.conf`

Description:

The `/etc/inetd.conf` file is the default configuration file for the *inetd* (p. 977) (super-server) daemon.

As shipped, this file contains commented-out lines that describe all currently supported QNX Neutrino TCP/IP daemons and some nonstandard *pidin* (p. 1521) services. If you want to use these daemons or services, edit `/etc/inetd.conf` and uncommon the appropriate lines. You can also add or remove daemon definitions if necessary.

The file must have an entry in each of its fields, with each field separated by a tab or a space. Comments are denoted by a pound sign (**#**) at the beginning of a line.

The fields in the configuration file are:

```
[addr:]service-name | service-name/version
socket-type
protocol[,sndbuf=size][,rcvbuf=size]
wait|nowait[:max]
user[:group]
server-program
server program arguments
```

Here's a description of the arguments:

addr

The local host address that `inetd` uses when listening for a service. (Not applicable to RPC-based services.) A single asterisk character (`*`) indicates that it is to listen on all local addresses (`INADDR_ANY`).

When a line contains an address specifier and colon only (no *service-name* field is specified), the address specifier is assumed for all further lines until another line with an explicit address specifier appears, or until the end of the file is reached.

service-name

Name of a valid service in the */etc/services* (p. 1744) file, or a valid RPC service name from */etc/rpc* (p. 1694).

Copyright © 2014, QNX Software Systems Limited

For “internal” services (see *server program arguments* (p. ?)), the service name must be the official name of the service (i.e. the first entry in `/etc/services`).

version

The RPC version number. It can simply be a single numeric argument or a range of versions. A range is bounded by the low version to the high version (e.g. `rusers/1-3`).

socket-type

One of `stream`, `dgram`, or `raw`, depending on whether the socket is a stream, datagram, or raw socket.

protocol

A valid protocol; for example, `tcp` or `udp` from `/etc/protocols`.

If you need to specify an IP version explicitly, use protocols such as `tcp4` (for IPv4) or `udp6` (for IPv6). Protocols, such as `tcp` or `udp`, default to the current IP version (currently IPv4).

For RPC-based services, you must prefix the protocol with `rpc/` (e.g `rpc/udp`).

`rcvbuf=`*size* or `sndbuf=`*size*

Size of the send or receive buffer for the listening socket. This may be useful for the TCP protocol because the window scale factor, that's based on the receive socket buffer size, is advertised when the connection handshake occurs. Therefore, the socket buffer size for the server must be set on the listening socket. In some situations, you may realize better TCP performances when increasing the socket buffer sizes. The socket buffer sizes are specified by appending their values to the protocol specification as follows:

```
tcp,rcvbuf=16384
tcp,sndbuf=64k
tcp,rcvbuf=64k,sndbuf=1m
```

A literal value may be specified or modified using `k` (for kilobytes) or `m` (for megabytes). Socket buffer sizes may be specified for all services and protocols except for the TCP port service multiplexer (TCPMUX) services.

`wait|nowait`

Tell `inetd` if it should wait for the server program to return, or to continue processing connections on the socket. Sockets other than datagram sockets should have a `nowait` entry in this space. If a datagram server connects to

Copyright © 2014, QNX Software Systems Limited

its peer, freeing the socket so `inetd` can receive further messages on the socket, it's said to be a *multi-threaded server* and should use the `nowait` entry.

If a datagram server processes all incoming datagrams on a socket and eventually times out, that server is said to be single-threaded and should use a `wait` entry. The `tftpd` daemon is an exception; it's a datagram server that establishes pseudo-connections. It must be listed as `wait` in order to avoid a race; the server reads the first packet, creates a new socket, and then forks and exits to let `inetd` check for new service requests to spawn new servers.

Stream servers are usually marked as `nowait`, but if a single server process is to handle multiple connections, it may be marked as `wait`. The master socket is passed as `fd 0` to the server, which then needs to accept the incoming connection. The server should eventually time out and exit when no more connections are active. The `inetd` daemon will continue to listen on the master socket for connections. The `identd` server is usually the only stream server marked as `wait`.

max

Maximum number of server instances that may be spawned from `inetd` within an interval of 60 seconds. If omitted, *max* defaults to 40 server instances.

user

Name of the user that the server runs as. This allows servers to be given less permission than `root`.

group

Allow servers to run with a different (primary) group ID than specified in the password file. If a group is specified and user isn't `root`, the supplementary groups associated with that user will still be set.

A group name is specified by appending a colon or dot (allowed for backwards compatibility) to the user name followed by the group name.

server-program

Pathname of the program that `inetd` executes when a request is found on `inetd`'s socket. If the desired service is provided internally by `inetd` (e.g. see `echo` in the *inetd* (p. 977) utility page), this field would contain the word `internal`.

server program arguments

Copyright © 2014, QNX Software Systems Limited

Any arguments to be passed to the server program. The name of the program is passed as *argv*[0]. If the *server program* field is `internal`, you can leave this field blank.

Setting the IPsec policy

You can specify the IPsec policy setting for each socket in a special comment line. A line that starts with the special comment "#@" identifies the policy specifier, and the content of the comment line is treated as the IPsec policy string.

Valid policy settings for `/etc/inetd.conf` include:

```
direction bypass
direction entrust
direction ipsec request ...
```

See "Setting the policy" in the IPsec protocols page for detailed descriptions of the arguments.

Multiple IPsec policy strings may be specified using semicolons as separators. If conflicting strings are found in a single line, the last string takes effect.

When a policy specifier is set with #@, all further lines in the `/etc/inetd.conf` configuration file are also affected. You can reset the IPsec policy by inserting a comment line without a policy string (i.e. a comment line containing #@ only).

If an invalid IPsec policy string appears in `/etc/inetd.conf`, `inetd` leaves error messages using *syslog()*, and terminates itself.

IPv6 TCP/UDP behavior

If you want to run a server for both IPv4 and IPv6 traffic, you'll need to run two separate processes for the same server program. You do this by adding two separate lines in `inetd.conf`, one for `tcp4` and one for `tcp6`.

Under various combination of IPv4/v6 daemon settings, `inetd` behaves as follows:

If you have:	IPv4 traffic:	IPv6 traffic:
Only one server on `tcp4`	Routed to the server	Isn't accepted
Two servers: one on `tcp4` and one on `tcp6`	Routed to the server on `tcp4`	Routed to the server on `tcp6`
Only one server on `tcp6`	For certain configurations, *may* be routed to the `tcp6` server (see the IP6 protocol page for details).	Routed to the server on `tcp6`.

Copyright © 2014, QNX Software Systems Limited

Examples:

The following is an example from a working `inetd.conf` file:

```
ftp  stream tcp nowait root  /usr/sbin/ftpd  in.ftpd -el
```

where:

`ftp`

Is the *service name* (see *`/etc/services`* (p. 1744)).

`stream`

Is the *socket type*.

`tcp`

Is the *protocol*.

`nowait`

Is the *wait/nowait* entry.

`root`

Is the *user*.

`/usr/sbin/ftpd`

Is the *server program*.

`in.ftpd`

Is `argv[0]` (*server program arguments*).

`-el`

Is `argv[1]` (*server program arguments*).

Copyright © 2014, QNX Software Systems Limited

inflator

Inflate previously deflated files

You must be `root` to start this resource manager.

Syntax:

```
inflator [-b num] [-t num] [-v[v...]] [mountpoints...]
         [!exclude...]
```

Runs on:

QNX Neutrino

Options:

-b *num*

The number of decompression buffers (default: 8).

-t *num*

The maximum number of worker threads; the default is 4.

-v[v...]

Be verbose. Each additional `v` makes the output more verbose.

If you specify the -v option, you need to start `inflator` in the background. If you don't use this option, `inflator` automatically daemonizes itself.

mountpoints

Directories to overlay (the default is `/`). If a mountpoint starts with an exclamation mark (`!`), that directory is excluded.

!*exclude*

Directories to exclude.

Copyright © 2014, QNX Software Systems Limited

Description:

The `inflator` resource manager sits in front of other filesystems and inflates files that were previously deflated using the *`deflate`* (p. 183) utility. It's typically used when the underlying filesystem is a flash filesystem — it can almost double the effective size of the flash memory.

When started without arguments, `inflator` takes over `/`, placing it in front of any existing filesystems. It then catches each *open()* first. If the file is being opened for read, `inflator` attempts to open the file itself on an underlying filesystem. It reads the first 16 bytes and checks for the signature of a deflated file. If the file was deflated, `inflator` places itself between the application and the underlying filesystem. All reads return the original file data before it was deflated.

From the application's point of view, the file appears to be uncompressed. Random seeks are also supported. If the application does a *stat()* on the file, the size of the inflated file (the original size before it was deflated) is returned. If it's necessary to open a deflated file and see the deflated data or the deflated size, append `.~~~` to the filename and open that. For example:

```
$ deflate file1    # Deflate a file
$ wc file1         # wc sees the contents of the original file
$ wc file1.~~~     # wc sees the contents of the deflated file
$ ls -l file1      # ls reports the size of the original file
$ ls -l file1.~~~  # ls reports the size of the deflated file
```

If a file opened for read doesn't have the deflate signature, `inflator` returns `ENOENT`, which gives the file to the next underlying filesystem. In this manner, `inflator` gets out of the way. Likewise if a file is opened for any write mode, `inflator` again returns `ENOENT` and gets out of the way. You can use the -v option to monitor the opens that `inflator` receives and which ones it accepts. You can use multiple -v options to obtain more verbosity.

By default, `inflator` maintains 8 inflation buffers that are used to decompress data. If more than 8 files are being processed at one time, the buffers are used as a cache. If a buffer is stolen away from a file, a subsequent read on that file requires the data to be reread and inflated again. A single -v prints a message each time a buffer is stolen, allowing you to tune your system. Since memory is often in short supply in embedded systems, you should use the minimum number of buffers needed to achieve acceptable performance.

The size of the buffers is determined by the `deflate` utility. It defaults to 8K, but may be set to any of 4K, 8K, 16K or 32K. In effect, the file is broken into smaller compression blocks instead of compressing the entire file as a whole. Without this, random seeking in the file would be a performance nightmare. On fast processors (200MHz), the cost of inflating the data shouldn't be significant. On slower processors, it may be a performance issue as the inflation code competes for processor cycles.

Copyright © 2014, QNX Software Systems Limited

Some versions of the flash filesystem (`devf-*`) also support compression. For most systems, `inflator` provides better compression and comparable, if not better, performance.

Examples:

Take over `/` and inflate files that have been deflated when they're opened for reading by applications:

```
inflator
```

Take over `/` as above, but pass on any files under `/tmp` to any underlying filesystems without examining them in any way:

```
inflator / !/tmp
```

Take over directories where executables are usually located. Inflate files that have been deflated when they're opened for reading by applications:

```
inflator /sbin /bin /usr/sbin /usr/bin
```

Take over `/` as above, but output some diagnostics (note the ampersand, which is needed to make `inflator` run in the background when -v is specified):

```
inflator -v &
```

Take over `/` as above, but output lots of diagnostics:

```
inflator -vvvvv &
```

Caveats:

You can't run `inflator` twice on the same mountpoint.

Copyright © 2014, QNX Software Systems Limited

infocmp

Output the contents of a terminfo capability file (UNIX)

Syntax:

```
infocmp [-1CILQ] [-w width] [terminal_name]
```

Runs on:

QNX Neutrino

Options:

-1

("one") Force the output to contain one entry per line. Without this option, output is printed to a maximum width of 60 characters, each line containing as many entries as can fit on that line.

-C

Generate output in `termcap` format.

-I

Generate output in `terminfo` format. This option is the default.

-L

Use the long C variable names listed in `/usr/include/term.h`

-Q

Use the names that map onto the QNX console capabilities.

-w *width*

Force the output to the specified number of columns. As many entries as fit on each line are output.

terminal_name

The name of a terminal capability file to display. If the ***TERMINFO*** environment variable is *not* set, the file is read from the `/usr/lib/terminfo` directory in a subdirectory named by the first letter of the terminal name. For example, the VT100 `terminfo` file is stored in: `/usr/lib/terminfo/v/vt100`. If set, the ***TERMINFO*** environment variable defines a directory where the compiled description file is read from.

Copyright © 2014, QNX Software Systems Limited

If a *terminal_name* isn't specified, the terminal named in the ***TERM*** environment variable is used.

Description:

The `infocmp` utility is used to output the contents of a previously compiled `terminfo` capability file in a number of formats. The default format is suitable for editing and recompilation with the *`tic`* (p. 1960) utility. The output is sorted such that the Boolean capabilities are output first, followed by the integer capabilities and then the string fields. The `infocmp` utility, executed without any options, produces the `terminfo` description of the currently defined terminal in a form suitable for editing and recompilation with the *`tic`* (p. 1960) utility.

If no options are specified and zero or one terminal name is specified, the -I option is assumed.

For more information, see Strang, John, Linda Mui, and Tim O'Reilly. 1988. *termcap & terminfo*. Sebastopol, CA: O'Reilly and Associates. ISBN 0937175226.

Copyright © 2014, QNX Software Systems Limited

io-audio

Start an audio driver

You must be `root` to start this manager.

Syntax:

```
io-audio [-d driver [opt[,opt...]]] [-m opt[,opt...]]
         [-o opt[,opt...]] [v[v]...]
```

Runs on:

QNX Neutrino

Options:

-d *driver* [*opt*[,*opt*...]]

Load the specified *driver* and pass it the given driver options. The *driver* name is the name of the shared object without the `deva-ctrl-` prefix and the `.so` extension. For example, to load the `deva-ctrl-4dwave.so` shared object, specify `4dwave` as the driver name.

All audio drivers support the following card options:

unit=*number*

The card number to mount the driver as.

dindex=*number*

The device number that additional following options apply to.

play_name=*name*

The symbolic name to assign to the PCM playback device.

cap_name=*name*

The symbolic name to assign to the PCM capture device.

You can start more than one driver by using multiple -d command-line options. For information on the drivers and any specific options, see the entries for the *deva-ctrl-** (p. 169) shared objects.

Copyright © 2014, QNX Software Systems Limited

-m ***opt*****[,*****opt*****...]**

Memory options. The *opt* variable can be any of the following:

pool_size=*kbytes*

The size of the DMA memory pool to create, in KB.

pool_name=*string*

The name of a shared memory object to map and use as DMA memory pool. This object must be physically contiguous memory.

-o ***opt*****[,*****opt*****...]**

Global options. The *opt* variable can be any of the following:

config_write_delay=*time*

The time in seconds after the last change before sound card settings are written to disk, (a value of -1 prevents the settings from ever being written).

data_thread_prio=*priority*

The priority of the software mixer thread. The default is 25.

disable_sw_mixer

On cards that have only a single channel in hardware, don't use software techniques to increase the maximum number of playing channels.

intr_thread_prio=*priority*

Set the priority of the interrupt service threads. The default is 50.

max_dma_buf_size=*size*

The maximum size, in kilobytes, for the DMA buffer.

sw_mixer_rate=[FAHQ|FA|L*value*]

Set the method of selecting the sampling frequency used by the PCM software mixing device if the underlying hardware device supports multiple rates:

- FAHQ (First Active High Quality) — select the highest HW-supported sample rate that best fits the first active clients requested rate (all subsequent subchannels will be locked to the active rate).

Copyright © 2014, QNX Software Systems Limited

- FA (First Active) — select the sample rate based on the First Active client's requested sample rate; all subsequent subchannels are locked to the active rate.
- L*value* — lock the sample rate to the rate specified by *value*, regardless of the client's requested rate.

The default is FAHQ.

sw_mixer_samples=*num*

Adjust the fragment size used by the software mixer to something other than the default of 2048 samples. Here's an example of how to calculate the value to pass in with this option:

sw_mixer_samples = samples_per_frag *
(hardware_rate/requested_rate) *
(hardware_voices/requested_voices)

-v

Increase the level of verbose output.

Description:

The `io-audio` manager provides support for dynamically loaded audio-driver modules. This utility enables you to load the audio drivers specified by the -d options when you start `io-audio`.

Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the *on* (p. 1417) command to adjust the priorities of the audio and graphics drivers.

Once `io-audio` has started, you can dynamically load and unload drivers using the `mount` and `umount` commands. E.g. this command:

```
io-audio -dvortex -daudiopci &
```

gives the same result as this sequence:

```
io-audio &
mount -T io-audio vortex
mount -T io-audio audiopci
```

When searching for shared objects, the `io-audio` manager uses the ***LD_LIBRARY_PATH*** environment variable.

Copyright © 2014, QNX Software Systems Limited

To unload a module, use a command like this:

```
umount /dev/snd/controlC0
```

Examples:

Provide support for Aureal Vortex sound card:

```
io-audio -vv -d vortex &
```

Load the AudioPCI driver, specifying the size of the DMA memory pool as 500 KB:

```
io-audio -vv -m pool_size=500 -d audiopci &
```

Lock the sample rate to 8 KHz (if the hardware supports 8 KHz natively):

```
io-audio -o sw_mixer_rate=L8000 -d my_audio_driver &
```

Start an audio driver, specifying the card options:

```
io-audio -d my_audio_driver \
cap_name=capture,play_name=playback_hw,dindex=1,play_name=playback_pcm_mixer
```

The `/dev/snd` directory will look something like this:

```
# ls -l /dev/snd
total 0
lrw-rw-rw-  1 root      root             0 May 31 11:11 capture -> pcmC0D0c
-rw-rw-rw-  1 root      root             0 May 31 11:11 controlC0
-rw-rw-rw-  1 root      root             0 May 31 11:11 mixerC0D0
-rw-rw-rw-  1 root      root             0 May 31 11:11 pcmC0D0c
-rw-rw-rw-  1 root      root             0 May 31 11:11 pcmC0D0p
-rw-rw-rw-  1 root      root             0 May 31 11:11 pcmC0D1p
lrw-rw-rw-  1 root      root             0 May 31 11:11 pcmPreferredc -> pcmC0D0c
lrw-rw-rw-  1 root      root             0 May 31 11:11 pcmPreferredp -> pcmC0D1p
lrw-rw-rw-  1 root      root             0 May 31 11:11 playback_pcm_mixer -> pcmC0D1p
lrw-rw-rw-  1 root      root             0 May 31 11:11 playback_hw -> pcmC0D0p
```

Files:

In addition to the *deva-ctrl-** (p. 169) drivers, the `io-audio` command can load the following shared objects:

deva-mixer-ac97.so **(p. 213)**

Mixer DLL for the AC97 codec.

deva-mixer-ak4531.so **(p. 214)**

Mixer DLL for the AK4531 codec.

deva-mixer-hda.so **(p. 215)**

Mixer DLL for High Definition Audio codecs.

deva-util-restore.so **(p. 216)**

Shared object used to restore an audio driver's state.

Copyright © 2014, QNX Software Systems Limited

io-blk.so

Block I/O support

Syntax:

driver [blk *option*[,*option*…]] [*fstype* [*options*]]

Runs on:

QNX Neutrino

Options:

The *driver* is one of the `devb-*` drivers, such as *devb-eide* (p. 235), and *option* is one of the options described below.

The optional *fstype* argument is one of the filesystem drivers (*fs-** (p. 727)); you can follow it with options specific to the filesystem.

Suffixes for size, memory, and time arguments

You can use suffixes on the arguments to some options to specify the units. These suffixes aren't case-sensitive.

For size arguments, the suffixes are:

- `b` — bytes
- `k` — kilobytes
- `m` — megabytes
- `%` — percent of the total amount of cache, etc., depending on the option

For memory arguments, the suffixes also include:

- `g` — gigabytes
- `p` — pages

For time arguments, the suffixes are:

- `h` — hours
- `m` — minutes
- `s` — seconds

`blk` options

You can specify the following options only in the `blk` section:

alloc=*mode*

Copyright © 2014, QNX Software Systems Limited

Set the cache/memory allocation policy to one of the following:

- `cache` — allocate all of the buffer cache (the cache=*size*) at startup, but allocate all other caches (e.g. names) on demand and let them grow to their specified limit, and then start removing the Least Recently Used (LRU) items.
- `demand` — allocate the buffer cache the same way, on-demand (it will grow from 0 to the size specified by the cache option as you access the disk).
- `upfront` — pregrow all caches to their full size. This option can be useful in RAM-tuning a system, to see how much memory the filesystem will eventually consume (things such as the name and vnode caches tend to grow over time).

The default is `cache`.

auto=*amount*

Set the amount of automounting to be performed; *amount* is one of:

- `none` — only raw block devices appear.
- `partition` — enumerate any partition tables.

The default is `partition`.

automount=*dev*[:*mountpoint*[:*fstype*[:*options*]]] or automount=@*filename*

Create a mountpoint for *dev* at *mountpoint*. If you don't specify a full path for the device, `io-blk.so` uses the value of its devdir option as a prefix. For example, if devdir is `/dev` (the default), an option of automount=hd0t77:/disk mounts `/dev/hd0t77` at `/disk`.

The optional *fstype* specifies the filesystem type, after which you can set options. The choices of filesystem and the associated shared objects are:

`cd`

`fs-cd.so` (p. 787)

`dos`

`fs-dos.so` (p. 795)

`ext2`

`fs-ext2.so` (p. 807)

`mac`

`fs-mac.so` (p. 809)

Copyright © 2014, QNX Software Systems Limited

nt

fs-nt.so (p. 818)

qnx4

fs-qnx4.so (p. 820)

qnx6

fs-qnx6.so (p. 823) (Power-Safe filesystem)

udf

fs-udf.so (p. 829)

If you don't specify the type of filesystem, the library tries to determine it automatically.

If the @*filename* version of this option is used, the automounts are as specified in the given file. The file is a list of mounts (using the same syntax as above), separated by newline characters or commas.

You can't locate the *filename* file in the filesystem to be automounted: it has to be available in an existing filesystem such as the image filesystem. Optionally, you could locate it in any devb filesystem that's already running.

To mount multiple filesystems on a (removable) device, specify that the device is shared with a + prefix. For example,

```
automount=+fd0:/dos/a:dos,automount=+fd0:/fd:qnx4
```

For a list of common partition types, see the Filesystems chapter of the *System Architecture* guide.

cache=*total*[:*hash*]

Specify the total size of the disk buffer cache. The buffer cache is used as intermediate storage for all disk I/O, as well as providing LRU caching for dirty delayed-write blocks and recently-read blocks.

By default, 15% of system RAM is assigned, with a minimum of 512 KB and a maximum of 512 MB. If you specify an explicit size, bounds of 512 KB and 3 GB are applied.

Normally this cache is allocated at startup; if you specify alloc=demand, or the initial allocation failed, then the cache is dynamically grown as required up to the specified size.

Copyright © 2014, QNX Software Systems Limited

The optional *hash* argument specifies the size of the buffer cache hash list. You can specify any of the *suffixes* (p. 993) described above. The default is 25% of the number of entries in the cache.

- The default cache size is excessive for *devb-fdc* (p. 244) and *devb-ram* (p. 258). You'll probably want to reduce it to the minimum:

```
devb-fdc blk cache=512k &
```

- By default in QNX Neutrino 6.5 and later, `io-blk.so` allocates the filesystem buffer cache (blk cache=) on affected ARM platforms from a global memory region (`SHMCTL_ANON | SHMCTL_GLOBAL`) to avoid the per-process 32 MB limitation. To override this and make the allocation from the normal `devb-*` process heap, specify blk memory=sysram.

---

delwri=*delay1*[:*delay2*[:*postpone*]]

Specify the delay time for write-behinds to the media. A dirty disk block may remain in the cache without being physically written to the disk, to improve performance. The default is up to 3 seconds (*delay1*) for fixed media, and 1 second (*delay2*) for removable media. For more information, see "*Controlling writing operations* (p. 1002)," below.

The *postpone* argument specifies the number of seconds for which to keep a dirty disk block in memory if it's being continuously modified, before physically writing it to the disk. This applies only to fixed media; for removable media, writes aren't delayed beyond the *delay2* period. By default, *postpone* is the same as *delay1*.

devdir=*path*

The directory in which `io-blk` presents the physical devices as block-special files. The default is `/dev`.

devno=*type*

Controls how major device numbers are requested; *type* is one of:

- `name` — use the name of the device (e.g. `hd`, `cd`).
- `class` — use the CAM class of the device (e.g. `direct`, `readonly`).
- `common` — use a single class for all block devices.

The default is `name`.

Copyright © 2014, QNX Software Systems Limited

enumpart=*order*

Set the order for enumerating disk partitions; one of the following:

- `forward` — enumerate slots 1 through 4, followed by any extended partitions.
- `reverse` — enumerate slots 4 through 1, followed by any extended partitions.
- `windows` — enumerate the active partition, followed by any extended partitions, and then non-booting primary partitions. For more information about this order, see *http://support.microsoft.com/kb/q51978/*.

The default is `forward`.

exclusive

Require/obtain exclusive access of the mount device. This means that when a filesystem is mounted on a partition, nothing else is allowed to open that raw partition until the filesystem is unmounted.

fdinfo=*mode*

Specify the storing of open file names for the *iofdinfo()* query. The options for *mode* are:

- `ncache` — try to reconstruct the file name from the contents of the directory name cache. Don't rely on this option to supply the names of all open files (a file's name is supplied only if all components of its pathname are in the name cache).
- `always` — store the name used in each *open()* call to ensure that this name is always available.
- `never` — never supply the name of an open file.

The default is `always`.

map=*size*[:*hash*]

Set the number of entries in a cache used to map translations from logical blocks to physical ones. If this option isn't specified, the size is based in the value of the vnode option.

The *hash* argument specifies the size of the associated hash list; the default is 1/6 of the number of entries in the map.

memory=*type1*[:*type2*[:*type3*[:*type4*]]]

Copyright © 2014, QNX Software Systems Limited

Specify the typed memory pool or pools to use. For example, memory=sysram&below4G:sysram says to try `sysram` with the `below4G` modifier, and if no such region exists, then try plain `sysram`. (The same option works on systems with more or less than 4 GB of RAM.)

- It's up to the startup to set up typed memory. Use `pidin syspage=asinfo` to see the list.
- Generally you don't need to specify the memory option, in which case `io-blk.so` uses the normal *mmap()* pool; but on a system with more than 4 GB of RAM, it's mandatory.
- You might have to quote this option, in order to prevent the shell from interpreting special characters such as an ampersand (&).

For more information about typed memory, see "Typed memory" in the Interprocess Communication (IPC) chapter of the *System Architecture* guide.

mfu=*segmentation*

Specify the MFU:MRU segmentation (typically as a percentage, but it can be a size). You can specify any of the *suffixes* (p. 993) described above. The default is a 50:50 split.

The first time a sector is accessed, it goes into the MRU (Most Recently Used) region; if it's accessed again, it goes into the MFU (Most Frequently Used). The oldest cache blocks are removed from either the MRU or MFU region, so as to preserve this ratio.

naming=*scheme*

Set the device/partition naming scheme. The default is `0#`. For more information, see "*Naming schemes* (p. 1003)," below.

ncache=*size*[:*hash*]

Specify a name cache of *size* entries. Using more name cache entries speeds up path/file lookups at the expense of memory. Setting the *size* to 0 disables name caching. If this option isn't specified, the size is determined from the vnode option.

The *hash* argument specifies the size of the associated hash list; the default is 1/6 of the size of the number of entries in the name cache.

priority=*prio*

Set the priority of periodic filesystem callouts. The default is 21.

Copyright © 2014, QNX Software Systems Limited

ra=*min*[:*max*]

Set the minimum and maximum sizes of the read-ahead buffers. You can specify any of the *suffixes* (p. 993) described above. The default minimum is the system page size; the default maximum is 64 times the system page size.

ramdisk=*size*[:*sector*]

Create an internal ramdisk device (`/dev/ramX`) of the specified size, with the specified sector size. The *size* and *sector* variables can use the *suffixes* (p. 993) described above. The sector size must be power of 2 in the range from 512 through 4096 bytes; the default is 512 bytes.

The initial contents of this memory device are unspecified, so you must format it before using it as a filesystem (for example, with *dinit* (p. 626) for a QNX 4 filesystem, or *mkqnx6fs* (p. 1274) for a Power-Safe filesystem).

rmvpoll=*period*

The polling period for removable media. The default is 0 seconds.

rmvto=*delay*

Specify a removable media timeout (default: 2 seconds). After the specified period of inactivity, a disk access prompts validation of the media with the driver; if the driver reports that the media has been changed, all data blocks and cached information for that device are discarded and relearned.

This option can take a value of none, which disables removable media relearning. This isn't very useful for real removable devices (e.g. CDs), but if your device is on-board SD, or USB that isn't removable but the driver is advertising it as such, you can disable the verification overheads.

thread=*max*[:*low*[:*high*]]

Set the thread pool parameters (maximum, low water, and high water). The default is `12:2:5`.

verbose[=*level*]

Be verbose. The output is sent to the system logger, *slogger* (p. 1807).

The optional *level* argument is a series of alphabetic characters that indicates the categories of event to log:

- b — bad blocks

Copyright © 2014, QNX Software Systems Limited

- `c` — configuration
- `d` — direct I/O
- `f` — fsys module (`fs-*`)
- `i` — input
- `o` — output
- `r` — removable
- `v` — virtual filesystem (VFS)

An option of blk verbose means all (`bcdfiorv`), blk verbose=io means input plus output, blk verbose=!r means everything except removable, and so on. The default is none.

vnode=*size*[:*max*]

Specify the number of vnode entries (filesystem-independent inodes) The default is 1024 entries. Up to *size* vnodes may be active. Vnodes remain in this cache when the corresponding file is closed, making subsequent opens faster.

The *max* argument allows a momentary large number files to be open at the same time; the cache tries to stay at *size* entries, but grows if needed up to *max* entries before giving an error of `ENFILE`. The default value of *max* is 3 times *size*.

Filesystem options

You can apply the following options globally (in the `blk` section) or to a specific filesystem (for example, in the `qnx4` section for a QNX 4 filesystem):

after

Mount the filesystem so that it's resolved after any other filesystems mounted at the same pathname (in other words, it's placed behind any existing mount). When you access a file, the system looks on this filesystem last, and only if the file wasn't found on any other filesystems.

before

Mount the filesystem so that it's resolved before any other filesystems mounted at the same pathname (in other words, it's placed in front of any existing mount). When you access a file, the system looks on this filesystem first.

commit=*level*

Set the committing level of the filesystem, which controls how dirty system/user blocks are written to disk. The *level* is one of `none`, `low`,

Copyright © 2014, QNX Software Systems Limited

`medium` (the default), and `high`. If it's `none`, all writes are time-delayed (as specified by the delwri option); at `high`, all writes are performed synchronously. For more information, see "*Controlling writing operations* (p. 1002)," below.

error=*action*

Set the action to perform when a `fs-*` filesystem module detects an internal error. The *action* is one of:

- `ebadfsys` — simply return EBADFSYS to the client.
- `mountro` — return EBADFSYS to the client and remount the affected filesystem as read-only.

The default is `ebadfsys`.

marking=*mode*

Set the filesystem-dirty marking behavior. The *mode* must be `none` or `mount` (the default). If marking is on, the filesystem is marked as being dirty when it's mounted, and it's marked as being clean when it's unmounted. The method of marking depends on the filesystem.

[no]atime

Update/don't update the file's directory entry if the only change is the access time. The noatime option isn't strict POSIX 1003.2 behavior, but it's faster.

[no]creat

Allow/don't allow files to be created on this filesystem.

[no]exec

Allow/don't allow file execution from this filesystem.

[no]lock

Lock/don't lock removable media. If locked, the medium is treated as fixed.

[no]rmv

Don't/do allow invalid mounts on removable media (re-insert).

[no]suid

Ignore/don't ignore the set-user ID bit on files in this filesystem.

ro

Mount all drives/filesystems as read-only.

rw

Copyright © 2014, QNX Software Systems Limited

Mount all drives/filesystems as read-write (if the physical media permit). This is the default.

For more information about the before and after options, see "Ordering mountpoints" in the Process Manager chapter of the *System Architecture* guide.

If you specify a filesystem option (e.g. `noatime`) on a block filesystem, and then you remount the filesystem (`mount -u`), the flag is ignored. The absence of the flag is interpreted as your asking for access time updates to be turned on. There's no way for the code in `io-blk` to determine if you wanted to use the default, and therefore didn't specify anything, or really did want access time updates to be turned on, and therefore didn't specify anything.

Similarly, if you mounted the filesystem as read-only and then remount it, the filesystem returns to its default setting.

To maintain the settings, specify the options again using the -o option for the mount command. For example:

```
mount -u -o noatime ...
```

Description:

The `io-blk.so` library provides block I/O support, as used by the *devb-** (p. 169), drivers, and loads filesystem drivers (*fs-** (p. 727)) as necessary.

The default values of the map and ncache options are based on the value of the vnode option. This arrangement lets you configure a system by specifying the cache size and the number of files, and letting the library set the other options.

Controlling writing operations

There are various types of writing operations:

Synchronous (SYNC)

Start immediately and wait for completion.

Asynchronous (ASYNC)

Start immediately but don't wait for completion.

Delayed (DELWRI)

Don't start until after a timeout period and then perform as asynchronous. The blk *delwri=* (p. ?) option controls the timeout for the delayed format; if you set this option to 0, a delayed writing operation is the same as asynchronous.

As required

Copyright © 2014, QNX Software Systems Limited

Write only if you have to.

The types of data include:

User

What you *read()* and *write()*.

Metadata

Things associated with *stat()*, such as times and IDs.

Filesystem

Things such as bitmaps, extents, etc.

If a file has no links, the "as required" form of write operation is used, never going to disk unless the buffer or cache is needed (since the file has no links, the data isn't expected to be accessible after a power failure). If you open a file with `O_SYNC`, the synchronous format is always used.

Otherwise, the blk *commit* (p. ?) level controls the type of write to use for each level of data:

commit=	Filesystem data	Metadata	User data
`none`	DELWRI	DELWRI	DELWRI
`low`	ASYNC	DELWRI	DELWRI
`medium`	SYNC	DELWRI	DELWRI
`high`	SYNC	SYNC	SYNC

If you specify commit=none, you lose all write ordering (both for single multiblock updates and multiple-user operations). Hence, your chances of a useful recovery following a power failure are poor. We recommend that you use this option only if you have a uninterruptible power supply (UPS), or if you don't mind using *dinit* (p. 626) on your filesystem as a recovery tool.

Calling *close()* might force a metadata update, but does nothing to the user data. Calling *fsync()* always forces out any delayed-write blocks for the file, and so is useful only when commit isn't high.

Naming schemes

You can use the naming=*scheme* option to specify the naming scheme to use for devices and partitions. The format of *scheme* is as follows:

`0`# (where 0 is any digit and sets the first/base number)

Copyright © 2014, QNX Software Systems Limited

The raw devices are named 0, 1, and so on, and partitions are named from the device with a `t` followed by the OS type of the partition (see "Partitions" in the Filesystems chapter of the *System Architecture* guide). For example, a QNX partition could be named `hd0t77`.

For duplicate partitions, a period (`.`) and sequence number are appended (e.g. `hd0t12`, `hd0t12.1`, and `hd0t12.2` for logical/extended DOS partitions). This is the QNX Neutrino naming scheme.

`0a` (actually any digit and any letter; these set the first/base name)

The raw devices are named 0,1,..., and partitions are named `a`, `b`, and so on (e.g. `/dev/hd0`, `/dev/hd0a`, `/dev/hd0b`, `/dev/hd0c`, and so on). The name doesn't indicate the OS type of the partitions, just the order in which they were found.

`a1` (actually any letter and any digit; these set the first/base name)

The raw devices are named `a`, `b`, and so on. Primary partitions are named 1, 2, 3, and 4; if you don't have four of them, the unused numbers are skipped. Any extended partitions are numbered without gaps from 5 (e.g. `/dev/hda`, `/dev/hda1`, `/dev/hda2`, `/dev/hda5`, and so on).

The name doesn't indicate the OS type of the partition, just its location. This is the Linux naming scheme.

The default naming scheme is `0#`.

Change to a different naming scheme at your own risk:

- Some system components could have hard-coded assumptions about disk names.
- Don't use a different scheme unless you're in control of the entire system.
- If you use a different scheme, you'll need some external knowledge about what filesystem to mount on a partition, because you won't have the `t`*XXX* naming hint.

Files:

`charset.so`

This library includes the character sets for Japanese, traditional Chinese, and standard Chinese (code pages 932, 936, and 950); `io-blk.so` loads this library if it needs to mount a FAT volume, CD, or DVD that's formatted in one of these locales.

Copyright © 2014, QNX Software Systems Limited

io-hid

Manager for human-interface devices (HID)

You must be `root` to start this manager.

Syntax:

```
io-hid [-n name] -d driver [driver_options]... [-v] [-V] &
```

Runs on:

QNX Neutrino

Options:

-d *driver* [*driver_options*]

Load the specified *driver* and pass it the given *driver_options*. For information on the drivers and their syntax and options, see the `devh-*` entries:

Driver	Shared object
`usb`	*`devh-usb.so`* (p. 337)
`ps2ser`	*`devh-ps2ser.so`* (p. 331)

-n *name*

Set the server name. The default is `/dev/io-hid/io-hid`.

-V

Display server version.

-v

Be verbose.

Description:

The `io-hid` manager provides support for input devices and input clients. You can load drivers when you start `io-hid` by specifying the -d command-line option.

Clients such as `devc-con` and `devi-hid` connect to `io-hid` and interact with human-interface devices through `io-hid`.

Copyright © 2014, QNX Software Systems Limited

You can start more than one driver by using multiple -d command-line options.

Once `io-hid` has started, you can dynamically load and unload modules using the *`mount`* (p. 1312) and *`umount`* (p. 2009) commands.

The `io-hid` manager uses the ***LD_LIBRARY_PATH*** environment variable when searching for the shared objects. If ***LD_LIBRARY_PATH*** is not set, or the shared object in question isn't in one of its directories, or you want to override the default, specify the full path in the `mount` command.

Examples:

Load USB HID devices, PS/2 mouse, serial mouse on COM1 directly, and a PS/2 keyboard:

```
io-hid -dusb -dps2ser \
ps2mouse:mousedev:msoft:uart,1:kbd:kbddev &
```

or use this sequence of commands to do the same thing:

```
io-hid &
mount -T io-hid devh-usb.so
mount -T io-hid devh-ps2ser.so \
ps2mouse:mousedev:msoft:uart,1:kbd:kbddev
```

Unload a module:

```
umount /dev/io-hid/devh-usb
```


Copyright © 2014, QNX Software Systems Limited

io-pkt-v4, io-pkt-v4-hc, io-pkt-v6-hc

Networking manager

Syntax:

```
io-pkt-variant [-d driver [driver_options]] [-i instance]
               [-P priority] [-p protocol [protocol_options]]
               [-t threads] [-v]
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`.

Runs on:

QNX Neutrino

Options:

-d *driver* [*driver_options*]

Start the specified *devn-** (p. 169) or *devnp-** (p. 169) driver:

- You can specify *driver* without the `devn-` or `devnp-` prefix or the `.so` extension. For example, to start the *devnp-i82544.so* (p. 432) driver, specify -d i82544. If you specify the driver this way, `io-pkt*` looks for a `devnp-` version first. If there isn't one, `io-pkt*` loads the legacy `io-net` (`devn-`) version, using a special "shim" layer, *devnp-shim.so* (p. 447).
- If you want to load a specific version of a driver, specify the full path of the module (e.g. `/lib/dll/devn-i82544.so`).

The *driver_options* argument is a list of driver-specific options that the stack passes to the driver.

Use commas, not spaces, to separate the options.

The stack processes various driver options; for more information, see "*Generic driver options* (p. 1013)," below.

-i *instance*

The stack instance number, which is useful if you're running multiple instances of `io-pkt`. The `io-pkt` manager will service mount requests of type `io-pkt`*X*, where *X* is the instance number. For example:

```
io-pkt-v4 -i1 -ptcpip prefix=/alt
mount -Tio-pkt1 /lib/dll/devnp-i82544.so
```

Copyright © 2014, QNX Software Systems Limited

-P ***priority***

The priority to use for `io-pkt`'s main thread. The default is 21.

-p ***protocol*** **[***protocol_options***]**

The protocol to start, followed by a list of protocol-specific options.

Use commas, not spaces, to separate the options.

The available protocols include:

Protocol	Module
`autoip`	*`lsm-autoip.so`* (p. 1145)
`pf-v4`	*`lsm-pf-v4.so`* (p. 1148) (for use with `io-pkt-v4` or `io-pkt-v4-hc`)
`pf-v6`	*`lsm-pf-v6.so`* (p. 1148) (for use with `io-pkt-v6-hc`)
`qnet`	*`lsm-qnet.so`* (p. 1149)
`slip`	*`lsm-slip.so`* (p. 1157) (for use with `io-pkt-v4` or `io-pkt-v4-hc`)
`tcpip`	The stack includes TCP/IP; you need to specify this protocol only if you want to pass additional parameters (e.g. prefix=) to it. For more information about the options, see below.

-S

Don't register a `SIGSEGV` handler to quiesce the hardware if a segmentation violation occurs. This can help with debugging if it isn't possible to get a backtrace to the original code that generated the `SIGSEGV` through the signal handler.

-t ***threads***

The number of processing threads to create. By default, one thread is created per CPU. These threads are the packet-processing threads that operate at Layer2 and may become the stack thread. For more information, see the Overview chapter of the QNX Neutrino Core Networking *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

-v

If any errors occur while loading drivers and protocols, `io-pkt` sends messages to *`slogger`* (p. 1807). If you specify this option, `io-pkt` also displays them on the console.

TCP/IP options

If you specify the -p tcpip protocol, the *protocol_options* list can consist of one or more of the following, separated by commas without whitespace:

bigpage_strict

If the value of the pagesize option is bigger than `sysconf(_SC_PAGE SIZE)`, it's used only for the `mbuf` and cluster pools unless you also specify this option, in which case the page size is used for all pools.

cache=*0*

Disable the caching of packet buffers. This should be needed only as a debugging facility.

confstr_monitor

Monitor changes to configuration strings, in particular `CS_HOSTNAME`. By default, `io-pkt` gets the hostname once at startup.

enmap

Prevent automatic stack mapping of `en`*XX* interface names to the actual interface names. By default, the stack automatically maps the first registered interface to `en0` (if a real `en0` isn't present), the second interface to `en1`, and so on, in order to preserve backwards compatibility with `io-net`-style command lines.

fastforward=*X*

Enable (1) or disable (0) fastforwarding path. This is useful for gateways. This option enables, and is enabled by, forward; to enable only forward, specify forward,fastforward=0.

forward

Enable forwarding of IPv4 packets between interfaces; this enables fastforward by default. The default is off.

forward6

(`io-pkt-v6-hc` only) Enable forwarding of IPv6 packets between interfaces; off by default.

Copyright © 2014, QNX Software Systems Limited

ipsec

(`io-pkt-v4-hc` and `io-pkt-v6-hc` only) Enable IPsec support; off by default.

mbuf_cache=*X*

As `mbufs` are freed after use, rather than returning them to the internal pool for general consumption, up to *X* `mbufs` are cached per thread to allow quicker retrieval on the next allocation.

mclbytes=*size*

The `mbuf` cluster size. A *cluster* is the largest amount of contiguous memory used by an `mbuf`. If the MTU is larger than a cluster, multiple clusters are used to hold the packet. The default cluster size is 2 KB (to fit a standard 1500-byte Ethernet packet).

Specifying the cluster size can improve performance; for more information, see "Jumbo packets and hardware checksumming" in the Network Drivers chapter of the QNX Neutrino Core Networking *User's Guide*.

pagesize=*X*

The smallest amount of data allocated each time for the internal memory pools. This quantum is then carved into chunks of varying size, depending on the pool.

pfil_ipsec

(`io-pkt-v4-hc` and `io-pkt-v6-hc` only) Run packet filters on packets before encryption. The default is to do it after encryption.

pkt_cache=*X*

As `mbuf` and cluster combinations are freed after use, rather than return them to the internal pool for general consumption, up to *X* `mbufs` and clusters are cached per thread to allow quicker retrieval on the next allocation.

pkt_typed_mem=*object*

Allocate packet buffers from the specified typed memory object. For example:

```
io-pkt -ptcpip pkt_typed_mem=ram/dma
```

prefix=*/path*

Copyright © 2014, QNX Software Systems Limited

The path to prepend to the traditional /dev/socket. The is useful when running multiple stacks. Clients can target a particular stack by using the ***SOCK*** environmental variable. For example:

```
io-pkt -ptcpip prefix=/alt
SOCK=/alt ifconfig -a
```

random

Use /dev/random as the source of random data. By default, io-pkt uses a builtin pseudo-random number generator.

recv_ctxt=*X*

Specify the size of the receive context buffer, in bytes. The default is 65536; the minimum is 2048.

reuseport_unicast

If using the SO_REUSEPORT socket option, received unicast UDP packets are delivered to all sockets bound to the port. The default is to deliver only multicast and broadcast to all sockets.

rx_prio=*X* or rx_pulse_prio=*X*

The priority for receive threads to use (the default is 21). A driver-specific priority option (if supported by the driver) can override this priority.

somaxconn=*X*

Specify the value of SOMAXCONN, the maximum length of the listen queue used to accept new TCP connections. The minimum is the value in <sys/socket.h>.

stacksize=*X*

Specify the size of each thread's stack, in bytes. The default is 4096.

threads_incr=*X*

If the supply of functional connections is exhausted, increment their number by this amount, up to the value of threads_max. The default is 25.

The term “threads” in the TCP/IP threads_* options is a misnomer; it really refers to functional TCP/IP connections or blocking operations (*read()*, *write()*, *accept()*, etc.). It has nothing to do with the number of threads running in io-pkt-*.

threads_max=*X*

Copyright © 2014, QNX Software Systems Limited

Specify the maximum number of functional TCP/IP connections that the stack can service simultaneously. The default is 200.

threads_min=*X*

Specify the minimum number of functional TCP/IP connections. The default is 15, and the minimum is 4.

timer_pulse_prio=*priority*

The priority to use for the timer pulse. The default is 21.

Description:

The `io-pkt` manager provides support for Internet domain sockets, Unix domain sockets, and dynamically loaded networking modules. It comes in several stack variants:

`io-pkt-v4`

An IPv4 memory-reduced variant that *doesn't* support:

- IPv6
- Crypto / IPSec
- 802.11 a/b/g Wi-Fi
- Bridging
- GRE / GRF
- Multicast routing
- Multipoint PPP

`io-pkt-v4-hc`

IPv4 version of the stack that has full encryption and Wi-Fi capability built in and includes hardware-accelerated cryptography capability (Fast IPsec).

`io-pkt-v6-hc`

IPv6 version of the stack (includes IPv4 as part of v6) that has full encryption and Wi-Fi capability, also with hardware-accelerated cryptography.

In order to use SSL connections, you must have started *`random`* (p. 1654) with the -t option.

After you've launched `io-pkt*`, you can use the *`mount`* (p. 1312) command to start drivers or load additional modules such as *`lsm-pf-v4.so`* *or* *`lsm-pf-v6.so`* (p. 1148). If you want to pass options to the driver, use the -o option *before* the name of the shared object. For example:

```
mount -T io-pkt -o mac=12345678 devnp-bge.so
```


Copyright © 2014, QNX Software Systems Limited

- You can use *umount* (p. 2009) to unmount legacy io-net drivers, but not io-pkt* drivers. Other drivers may allow you to detach the driver from the stack, by using *ifconfig* (p. 957)'s destroy command (if the driver supports it).
- If io-pkt runs out of threads, it sends a message to *slogger* (p. 1807), and anything that requires a thread blocks until one becomes available.

- Native io-pkt and ported NetBSD drivers don't put entries into the /dev/io-net namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of waitfor /dev/io-net/en0, use if_up -p en0.
- If a TCP/IP packet is smaller than the minimum Ethernet packet size, the packet may be padded with random data, rather than zeroes.

The io-pkt manager supports TUN and TAP. To create the interfaces, use ifconfig:

```
ifconfig tun0 create
ifconfig tap0 create
```

For more information, see the NetBSD documentation:

- *http://netbsd.gw.com/cgi-bin/man-cgi?tun++NetBSD-current*
- *http://netbsd.gw.com/cgi-bin/man-cgi?tap++NetBSD-current*

Generic driver options

The stack processes the following generic driver options:

name=*prefix*

Override the default interface prefix used for network drivers. For example:

```
io-pkt-v4 -di82544 name=en
```

starts the devnp-i82544.so driver with the io-net-style interface naming convention (en*XX*). You can also use this option to assign interface names based on (for example) functionality:

```
io-pkt-v4 -di82544 pci=0,name=wan
```

unit=*number*

The interface number to use. If *number* is negative, it's ignored. By default, the interfaces are numbered starting at 0.

Copyright © 2014, QNX Software Systems Limited

These options don't work with legacy `io-net` legacy drivers. If you attempt to use them with a `devn-` driver, the driver won't be loaded, and the log will include an "unknown option" error.

The stack also processes the following driver options for all USB drivers using the NetBSD-to-QNX conversion library to let you identify a particular USB device using information obtained from running *usb -v* (p. 2025):

did=*ID*

Device product ID.

vid=*ID*

Device vendor ID.

devno=*addr*

Device address, as reported by the `usb` utility.

busno=*num*

Host controller, as reported by the `usb` utility

For example:

```
io-pkt-v4-hc -drum did=0x0020,vid=0x13b1,devno=1,busno=1
```


Copyright © 2014, QNX Software Systems Limited

io-usb

Host Controller Driver (HCD) server for universal serial bus (USB)

You must be `root` to start this server.

Syntax:

```
io-usb [-CcVv] [-d dll [opts] ] [-e priority]
       [-h num] [-m num] [-n name] [-P priority]...
       [-r num] [-t memory=name]
```

Runs on:

QNX Neutrino

Options:

-C

Never select a configuration at enumeration time (hubs excepted).

-c

Don't select a device configuration if the device has more than one configuration. See "*Selecting a driver configuration* (p. 1017)" below.

-d *dll* [*opts*]

Load the specified host controller DLL and *opts* and pass it the *dll*. For information on the drivers and their syntax and options, see the `devu-*` entries.

-e *priority*

Set the priority of the enumeration thread.

-h *num*

The high watermark for the thread pool. The default is 4.

-m *num*

The maximum number of threads in the thread pool. The default is 8.

-n *name*

Set the server name. The default is `/dev/io-usb/io-usb`.

Copyright © 2014, QNX Software Systems Limited

-P ***priority***

The priority to use for the server; the default is 21.

-r ***num***

Set the number of enumeration retries (default 3).

-t memory=*name*

Set the typed-memory name (default none, using sysram). This option tells the stack to allocate memory from the specified typed-memory region (such as the memory used for *usbd_alloc()*, and other internal memory that's needed to transfer data from the USB chip).

-V

Display the server version and then exit.

-v

Be verbose.

Description:

The `io-usb` Host Controller Driver (HCD) server contains USB protocols and communicates with clients (class drivers). The USB stack is a server/dll interface that the server uses to load the DLLs that manage the USB chips. You can load drivers when you start `io-usb` by specifying the -d command-line option.

- This is the host-side stack; for the device-side stack, see *`io-usb-dcd`* (p. 1018).
- You can start more than one driver by using multiple -d command-line options.

Once `io-usb` has started, you can dynamically load and unload modules using the *`mount`* (p. 1312) and *`umount`* (p. 2009) commands.

The `io-usb` controller uses the ***`LD_LIBRARY_PATH`*** environment variable when searching for the shared objects. If ***`LD_LIBRARY_PATH`*** is not set, or the shared object in question isn't in one of its directories, or you want to override the default, specify the full path in the `mount` command.

For example, to mount the EHCI (high speed) USB driver:

```
mount -Tio-usb devu-ehci.so /dev/io-usb/io-usb
```


Copyright © 2014, QNX Software Systems Limited

To mount the OHCI (full/low speed) USB driver:

```
mount -Tio-usb devu-ohci.so /dev/io-usb/io-usb
```

To mount the UHCI (full/low speed) USB driver:

```
mount -Tio-usb devu-uhci.so /dev/io-usb/io-usb
```

To mount the XHCI (high/full/low speed) USB driver:

```
mount -Tio-usb devu-xhci.so /dev/io-usb/io-usb
```

Selecting a driver configuration

You should use the -c option in conjunction with a launcher application, such as `usblauncher` (see the *Device Publishers Developer's Guide*), which choose a driver's configuration before launching the driver to manage a device's interfaces.

A launcher application must choose a default configuration for devices with more than one configuration, or these devices will not function:

- If you specify the -c option, `io-usb` will *not* select a device's configuration, leaving the configuration selection to the launcher application.
- If you do *not* specify the -c option, `io-usb` will automatically select the device's first configuration.

Some devices may not be able to switch the configuration once an initial configuration is selected.

Examples:

Start the USB 2.0 stack and USB drivers:

```
io-usb -dehci -dohci -duhci -dxhci
```

or use this sequence of commands to do the same thing:

```
io-usb &
mount -T io-usb devu-ehci.so /dev/io-usb/io-usb
mount -T io-usb devu-ohci.so /dev/io-usb/io-usb
mount -T io-usb devu-uhci.so /dev/io-usb/io-usb
mount -T io-usb devu-xhci.so /dev/io-usb/io-usb
```

Unload a module:

```
umount /dev/io-usb/devu-ehci.so
```

Copyright © 2014, QNX Software Systems Limited

io-usb-dcd

Device Controller Driver (DCD) server for universal serial bus (USB)

You must be `root` to start this server.

Syntax:

```
io-usb-dcd [-d dll [opts] ] [-n name] [-P priority]... [-V]
[-v]
```

Runs on:

QNX Neutrino

Options:

-d *dll* [*opts*]

Load the specified device controller DLL and pass the options (*opts*) to it. For information on the drivers and their syntax and options, see the `devu-*` entries.

-n *name*

Set the server name. The default is `/dev/io-usb-dcd/io-usb`.

-P *priority*

The priority to use for the server; the default is 21.

-V

Display the server version and then exit.

-v

Be verbose.

Description:

The `io-usb-dcd` Device Controller Driver (DCD) server contains USB protocols and communicates with clients (function drivers). The USB stack is a server/dll interface that the server uses to load the DLLs that manage the USB chips. You can load drivers when you start `io-usb-dcd` by specifying the -d command-line option.

Copyright © 2014, QNX Software Systems Limited

- This is the device-side stack; for the host-side stack, see *io-usb* (p. 1015).
- You can start more than one driver by using multiple -d command-line options.

Once `io-usb-dcd` has started, you can dynamically load and unload modules using the *mount* (p. 1312) and *umount* (p. 2009) commands.

The `io-usb-dcd` controller uses the ***LD_LIBRARY_PATH*** environment variable when searching for the shared objects. If ***LD_LIBRARY_PATH*** is not set, or the shared object in question isn't in one of its directories, or you want to override the default, specify the full path in the `mount` command.

Examples:

```
# Create an 8 MB RAM disk:
devb-ram ram capacity=16384,nodinit blk cache=0m

# Format the RAM disk as a DOS filesystem (substitute
# the /dev/hd entry created by devb-ram):
mkdosfs /dev/hdn

# Start the USB stack:
io-usb-dcd -dusbumass-$(HW_VARIANT) ioport=ioport,irq=irq

# Load the mass-storage function driver:
devu-umass_client-block -l lun=0,devno=1,iface=0,fname=/dev/hdn

# Turn on the link (enables connection to the USB host):
ulink_ctrl -l 1
```

Copyright © 2014, QNX Software Systems Limited

Chapter 11
J

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “J”.

Copyright © 2014, QNX Software Systems Limited

join

Merge sorted files (POSIX)

Syntax:

```
join [-1 n] [-2 n]
     [-a file_number] [-e string]
     [-j file_number n]
     [-o list] [-t char] [-v file_number]
     file1 file2
```

Runs on:

QNX Neutrino

Options:

Options -a, -j, and -v use the *file_number* argument. Specify 1 to refer to *file1*, or 2 to refer to *file2*.

-1 *n*

(One) Join on the *n*th field of *file1*. Fields are numbered, starting with 1.

-2 *n*

Join on the *n*th field of *file2*. Fields are numbered, starting with 1.

-a *file_number*

In addition to the default output, produce a line for every unpairable line in file *file_number*. If both -a 1 and -a 2 are specified, both sets of information are output, and information for -a 2 is always printed first.

-e *string*

Replace empty output fields by the string *string*.

-j *file_number n*

(Obsolescent) Join on the *n*th field of file *file_number*. If *file_number* is neither 1 nor 2 (e.g. -j 0 3), use the *n*th field in both files. Fields are numbered, starting with 1.

Use options -1 and -2 in place of option -j.

-o *list*

Copyright © 2014, QNX Software Systems Limited

Produce output in which each line comprises the fields specified in *list*. Each entry in *list* has the form:

file_number.field

where *field* is a field number. You can use a space or a comma to separate entries.

Output is written only for lines with matching join fields. The join field isn't written unless it's included in *list*.

-t *char*

Use the character *char* as a separator, for both input and output. Every appearance of *char* in a line is significant. When this option is specified, the collating sequence should be the same as that produced by *sort* (p. 1826), *without* the -b option.

-v *file_number*

Instead of the default output, produce a line only for every unpairable line in *file_number*.

file1 file2

The names of the text files. If either *file1* or *file2* is -, the standard input is used.

Description:

The `join` utility forms a "join" of the two relations specified by the lines of *file1* and *file2*. The join is written to the standard output.

The files *file1* and *file2* are compared on the basis of a "join field" found in both files. For every pair of lines in *file1* and *file2* that have identical join fields, `join` prints one output line. The output line normally consists of the join field, followed by the rest of the line from *file1* and then the rest of the line from *file2*. By default, the join field is the first field in each line.

Both *file1* and *file2* should be sorted in an increasing collating sequence on their join fields (i.e. the same sequence performed by *sort -b* (p. 1826)). Otherwise, some field matches may not be reported. Note, however, that when option -t is specified, the collating sequence should be the same as that produced by `sort`, *without* the -b option.

The default input field separators are blanks. In this case, multiple separators count as one field separator; leading separators are ignored. The default output field separator is a space.

Copyright © 2014, QNX Software Systems Limited

Examples:

Join the password file and group file, matching on the numeric group ID and outputting the login name, group name, and login directory. It's assumed that the files have been sorted in collating sequence on the group ID fields.

```
join -1 4 -2 3 -o 1.1 2.1 1.6 -t:  /etc/passwd /etc/group
```

Exit status:

0

All input files were output successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 12
K

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “K”.

Copyright © 2014, QNX Software Systems Limited

kill

Terminate or signal processes (POSIX)

Syntax:

```
kill [-a] [-n node] [-signal_name | -signal_number] pid...
kill -l
```

Runs on:

QNX Neutrino

Options:

-signal_name

The name of the signal to be sent to the specified process. Values of *signal_name* are recognized in case-independent fashion, with or without the `SIG` prefix.

-signal_number

A nonnegative decimal integer representing the signal to be sent to the specified process.

-a

(QNX Neutrino extension) Interpret the *pid* argument as the ID of the application to be signalled.

-l

("el") Don't send signals. List possible values for *signal_name*.

-n *node*

(QNX Neutrino extension) Kill the process on the specified node. This option isn't available in the shell builtin version of `kill`.

pid

A decimal integer specifying a process or process group to be signaled. A positive number *pid* is a process ID. A negative number *pid* has its absolute value taken as a process group ID. The signal is sent to all processes belonging to the group.

Copyright © 2014, QNX Software Systems Limited

A *pid* of zero sends the signal to all processes owned by the user in the current shell's process group.

Don't use a *pid* of zero when logged on as `root`. Signalling all the background processes owned by the superuser and any current superuser foreground processes (e.g., backups in progress) may produce unpredictable results.

If you specify the -a, `kill` signals all the processes whose application ID matches this argument (which must be positive).

Description:

The `kill` utility sends a signal to the process(es) specified by each *pid* operand. By default, `kill` sends the `SIGTERM` signal, but you may override this default by naming a signal to be sent.

To print the list of signals that may be sent, use `kill` with the -l option:

```
kill -l
```

The `kill` command is available both as a standalone utility and as a shell builtin. To make sure that you're using the utility, specify the full path. For information about the builtin command, see *esh* (p. 710) and *ksh* (p. 1065).

Examples:

Any of the commands:

```
kill -9 100 -16
kill -sigkill 100 -16
kill -KILL 100 -16
```

sends the `SIGKILL` signal to the process whose process ID is 100 and to all processes whose process group ID is 16, assuming the sending process has permission to send that signal to the specified processes, and that they exist.

Exit status:

0

At least one matching process was found for each *pid* option, and the specified signal was successfully sent to each matching process.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Caveats:

Some shells include a builtin `kill` command. To make sure you're using the `kill` utility, specify the the full path to it.

Copyright © 2014, QNX Software Systems Limited

ksh

Public domain Korn shell command interpreter (UNIX)

Syntax:

```
ksh [+-abCefhikmnprsuvxX] [+-o option]
    [ [ -c command-string [command-name]
      | -s | file ] [argument…] ]
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

You can specify the following options only on the command line:

-c *command-string*

Execute the command(s) contained in *command-string*.

-i

Use interactive mode.

-l

Login shell; also implies interactive mode.

-s

The shell reads commands from standard input; all non-option arguments are positional parameters.

-r

Use restricted mode.

In addition to the above, you can use the options described for the *set* (p. 1067) builtin command on the command line.

Description:

The ksh is a public-domain version of the Korn shell. It's a command interpreter that's intended for both interactive and shell script use.

This shell isn't the same as the standard QNX 4 shell, which was modeled after ksh86.

Copyright © 2014, QNX Software Systems Limited

This description includes the following sections:

Shell startup

If neither the -c nor the -s option is specified, the first non-option argument specifies the name of a file the shell reads commands from; if there are no non-option arguments, the shell reads commands from standard input. The name of the shell (i.e., the contents of the `$0` parameter) is determined as follows: if the -c option is used and there's a non-option argument, it's used as the name; if commands are being read from a file, the file is used as the name; otherwise the name the shell was called with (i.e., *argv*[0]) is used.

A shell is *interactive* if the -i option is used or if both standard input and standard error are attached to a tty. An interactive shell has job control enabled (if available), ignores the `SIGINT`, `SIGQUIT` and `SIGTERM` signals, and prints prompts before reading input (see ***PS1*** and ***PS2*** parameters). For noninteractive shells, the trackall option is on by default (see the *`set`* (p. 1067) command below).

A shell is *restricted* if the -r option is used or if either the basename of the name the shell is invoked with or the ***SHELL*** parameter match the pattern `*r*sh` (e.g., `rsh`, `rksh`, and so on). The following restrictions come into effect after the shell processes any profile and ***$ENV*** files:

- The *`cd`* (p. 1061) builtin command is disabled.
- The ***SHELL***, ***ENV*** and ***PATH*** environment variables can't be changed.
- Command names can't be specified with absolute or relative paths.

Copyright © 2014, QNX Software Systems Limited

- You can't use the -p option of the *command* (p. 1062) builtin command.
- Redirections that create files can't be used (i.e., >, >>|, >>, <>).

A shell is *privileged* if the -p option is used or if the real userid or group ID doesn't match the effective userid or group ID (see *getuid()*, and *getgid()*). A privileged shell doesn't process $***HOME***`/.profile` or the ***ENV*** environment variable (see below); instead it processes the `/etc/suid_profile` file. Clearing the privileged option causes the shell to set its effective userid (group ID) to its real userid (group ID).

If the basename of the name the shell is called with (i.e., *argv*[0]) starts with `-` or if the -l option is used, the shell is assumed to be a login shell, and the shell reads and executes the contents of `/etc/profile` and $***HOME***`/.profile` if they exist and are readable.

If the ***ENV*** environment variable is set when the shell starts (or, in the case of login shells, after any profiles are processed), its value is subjected to parameter, command, arithmetic and tilde substitution and the resulting file (if any) is read and executed. If ***ENV*** isn't set (and not null) and `ksh` was compiled with the DEFAULT_ENV macro defined, the file named in that macro is included (after the above mentioned substitutions have been performed).

The exit status of the shell is 127 if the command file specified on the command line couldn't be opened, or nonzero if a fatal syntax error occurred during the execution of a script. In the absence of fatal errors, the exit status is that of the last command executed, or zero, if no command is executed.

Command syntax

The shell begins parsing its input by breaking it into *words*. Words, which are sequences of characters, are delimited by unquoted whitespace characters (space, tab and newline) or meta-characters (<, >, `|`, `;`, &, `(` and `)`). Aside from delimiting words, spaces and tabs are ignored, while newlines usually delimit commands. The meta-characters are used in building the following tokens:

<, <&, <<, >, >&, >>, and so on.

Specify redirections (see "*Input/output redirection* (p. 1050)," below).

|

Creates pipelines.

|&

Creates coprocesses (see "*Coprocesses* (p. 1054)," below).

;

Separates commands.

&

Copyright © 2014, QNX Software Systems Limited

Creates asynchronous pipelines.

&& and ||

Specify conditional execution.

;;

Used in `case` statements.

((..))

Used in arithmetic expressions.

(..)

Create subshells.

You can quote whitespace and meta-characters individually using backslash (\), or in groups using double (") or single (') quotes. Note that the following characters are also treated specially by the shell and you must quote them if they're to represent themselves:

\, ", '

Quoting characters.

#

If used at the beginning of a word, introduces a comment — everything after the # up to the nearest newline is ignored.

An exception occurs when # is followed by `!`*shell*. This allows you to run a script using an alternative shell interpreter. For example, if you have a C shell in `/bin/csh`, you can tell `ksh` to run your scripts using the C shell by starting the scripts with this:

```
#!/bin/csh
```

$

Introduces parameter, command and arithmetic substitutions (see "*Substitution* (p. 1039)," below).

`

Introduces an old-style command substitution (see "*Substitution* (p. 1039)," below).

~

Begins a directory expansion (see "*Tilde expansion* (p. 1047)," below);

Copyright © 2014, QNX Software Systems Limited

{ and }

Delimit `csh`-style alternations (see "*Brace expansion* (p. 1048)," below).

***, ? and [**

Used in filename generation (see "*Filename patterns* (p. 1048)," below).

As words and tokens are parsed, the shell builds commands, of which there are two basic types: *simple commands*, typically programs that are executed, and *compound commands*, such as `for` and `if` statements, grouping constructs and function definitions.

A simple command consists of some combination of parameter assignments (see "*Parameters* (p. 1040)," below), input/output redirections (see "*Input/output redirection* (p. 1050)," below), and command words; the only restriction is that parameter assignments come before any command words. The command words, if any, define the command that's to be executed and its arguments. The command may be a shell builtin command, a function or an *external command*, i.e., a separate executable file that's located using the ***PATH*** parameter (see "*Command execution and builtin commands* (p. 1058)," below). Note that all command constructs have an *exit status*:

- For external commands, this is related to the status returned by *wait()* (if the command couldn't be found, the exit status is 127, if it couldn't be executed, the exit status is 126).
- The exit status of other command constructs (builtin commands, functions, compound commands, pipelines, lists, and so on) are all well defined and are described where the construct is described.

The exit status of a command consisting only of parameter assignments is that of the last command substitution performed during the parameter assignment, or zero if there were no command substitutions.

The ? special parameter stores the exit status of the last nonasynchronous command. For example, you can display the exit status by typing:

```
echo $?
```

For more information, see "*Parameters* (p. 1040)," below.

You can chain commands together using the `|` token to form *pipelines* in which the standard output of each command but the last is piped (see *pipe()*) to the standard input of the following command. The exit status of a pipeline is that of its last command. You can prefix a pipeline with the `!` reserved word, which causes the exit status of the pipeline to be logically complemented: if the original status is 0 the complemented status is 1, and if the original status isn't 0, then the complemented status is 0.

Copyright © 2014, QNX Software Systems Limited

You can create *lists* of commands by separating pipelines by any of the following tokens:

&&, ||

Conditional execution: *cmd1* && *cmd2* executes *cmd2* only if the exit status of *cmd1* is zero; || is the opposite — *cmd2* is executed only if the exit status of *cmd1* is nonzero.

The && and || tokens have equal precedence, which is higher than that of &, |& and ;, which also have equal precedence.

&

Causes the preceding command to be executed asynchronously, that is, the shell starts the command, but doesn't wait for it to complete (the shell does keep track of the status of asynchronous commands — see "*Job control* (p. 1082)", below). When an asynchronous command is started when job control is disabled (i.e., in most scripts), the command is started with signals INT and QUIT ignored and with input redirected from /dev/null (however, redirections specified in the asynchronous command have precedence).

|&

Starts a *coprocess*, which is special kind of asynchronous process (see "*Coprocesses* (p. 1054)", below).

;

Sequential execution: *cmd1* ; *cmd2* executes *cmd1*, and then executes *cmd2* when the first command is done.

Note that a command must follow the && and || operators, while a command need not follow &, |& and ;. The exit status of a list is that of the last command executed, with the exception of asynchronous lists, for which the exit status is 0.

Compound commands

Compound commands are created using the following reserved words. These words are recognized only if they're unquoted and are used as the first word of a command (i.e., you can't put any parameter assignments or redirections before them):

```
case    else    function    then    !
do      esac    if          time    [[
done    fi      in          until   {
elif    for     select      while   }
```

Some shells (but not this one) execute control structure commands in a subshell when one or more of their file descriptors are redirected, so any environment changes inside them may fail. To avoid problems with portability, you should

Copyright © 2014, QNX Software Systems Limited

use the *exec* (p. 1063) statement instead to redirect file descriptors before the control structure.

In the following compound command descriptions, command lists (denoted as *list*) that are followed by reserved words must end with a semicolon, a newline or a (syntactically correct) reserved word. For example, these are valid:

```
{ echo foo; echo bar; }
{ echo foo; echo bar<newline>}
{ { echo foo; echo bar; } }
```

but this isn't:

```
{ echo foo; echo bar }
```

The commands are:

(*list*)

Execute *list* in a subshell. There's no implicit way to pass environment changes from a subshell back to its parent.

{ *list* }

Compound construct; *list* is executed, but not in a subshell. Note that { and } are reserved words, not meta-characters.

`case` *word* `in [ [(]` *pattern* `[| pattern]…)` *list* `;; ] …` `esac`

The `case` statement attempts to match *word* against the specified *patterns*; the *list* associated with the first successfully matched pattern is executed.

Patterns used in `case` statements are the same as those used for filename patterns, except that the restrictions regarding `.` and `/` are dropped. Note that any unquoted space before and after a pattern is stripped; you must quote any space within a pattern. Both the word and the patterns are subject to parameter, command, and arithmetic substitution as well as tilde substitution.

For historical reasons, you can use open and close braces instead of `in` and `esac` (e.g., `case $foo { *) echo bar; }`).

The exit status of a `case` statement is that of the executed *list*; if no *list* is executed, the exit status is zero.

`for` *name* `[ in` *word* `…` *term* `]` `do` *list* `done`

For each *word* in the specified word list (where *term* is either a newline or a `;`), the parameter *name* is set to the word and *list* is executed. If `in` isn't

Copyright © 2014, QNX Software Systems Limited

used to specify a word list, the positional parameters (`$1`, `$2`, and so on) are used instead.

For historical reasons, you can use open and close braces instead of `do` and `done` (e.g., `for i; { echo $i; }`).

The exit status of a `for` statement is the last exit status of *list*; if *list* is never executed, the exit status is zero.

`if` *list* `then` *list* `[elif` *list* `then` *list*`]` … `[else` *list*`] fi`

If the exit status of the first *list* is zero, the second *list* is executed; otherwise the *list* following the `elif`, if any, is executed with similar consequences. If all the lists following the `if` and `elif` clauses fail (i.e., exit with nonzero status), the *list* following the `else` is executed.

The exit status of an `if` statement is that of the nonconditional *list* that's executed; if no nonconditional *list* is executed, the exit status is zero.

`select` *name* `[ in` *word* … *term* `]` `do` *list* `done`

The `select` statement provides an automatic method of presenting the user with a menu and selecting from it. An enumerated list of the specified *words* (where *term* is either a newline or a `;`) is printed on standard error, followed by a prompt (***PS3***, normally `"#? "`). A number corresponding to one of the enumerated words is then read from standard input, *name* is set to the selected word (or is unset if the selection isn't valid), ***REPLY*** is set to what was read (leading/trailing spaces are stripped), and *list* is executed.

If you enter a blank line (i.e., zero or more `IFS` characters), the menu is reprinted without executing *list*. If ***REPLY*** is null when *list* completes, the enumerated list is printed, the prompt is printed and so on. This process is continues until an end-of-file is read, an interrupt is received or a break statement is executed inside the loop. If `in` *word* … is omitted, the positional parameters are used (i.e., `$1`, `$2`, and so on).

For historical reasons, you can use open and close braces instead of `do` and `done` (e.g., `select i; { echo $i; }`).

The exit status of a `select` statement is zero if a break statement is used to exit the loop, nonzero otherwise.

`until` *list* `do` *list* `done`

This works like `while`, except that the body is executed only while the exit status of the first *list* is nonzero.

`while` *list* `do` *list* `done`

Copyright © 2014, QNX Software Systems Limited

A `while` is a prechecked loop. Its body is executed as often as the exit status of the first *list* is zero. The exit status of a `while` statement is the last exit status of the *list* in the body of the loop; if the body isn't executed, the exit status is zero.

`function` *name* `{` *list* `}`

Define the function *name*. See "*Functions* (p. 1055)," below. Note that redirections specified after a function definition are performed whenever the function is executed, not when the function definition is executed.

name* `()` *command

Mostly the same as `function`. See "*Functions* (p. 1055)," below.

`((` *expression* `))`

Evaluate the arithmetic expression *expression*; equivalent to `let "`*expression*`"`. See "*Arithmetic expressions* (p. 1052)" and the *`let`* (p. 1065) builtin command below.

`[[` *expression* `]]`

Similar to the *`test` and `[ ... ]`* (p. 1071) builtin commands (described later), with the following exceptions:

- Field splitting and filename generation aren't performed on arguments.
- The `-a` (and) and `-o` (or) operators are replaced with `&&` and `||`, respectively.
- You must leave operators (e.g., `-f`, `=`, `!`, and so on) unquoted.
- The second operand of `!=` and `=` expressions are patterns (e.g., the comparison in `[[ foobar = f*r ]]` succeeds).
- There are two additional binary operators, `<` and `>` that return true if their first string operand is less than or greater than their second string operand.
- The single argument form of `test`, which tests if the argument has nonzero length, isn't valid; you must always use explicit operators. For example, instead of:

  ```
  [ str ]
  ```

  use:

  ```
  [[ -n str ]]
  ```

Copyright © 2014, QNX Software Systems Limited

- Parameter, command and arithmetic substitutions are performed as expressions are evaluated and lazy expression evaluation is used for the && and || operators. This means that in the statement:

  ```
  [[ -r foo && $(< foo) = b*r ]]
  ```

  the `$(< foo)` is evaluated if and only if the file `foo` exists and is readable.

Quoting

Quoting is used to prevent the shell from treating characters or words specially:

- A backslash (\) quotes the following character, unless it's at the end of a line, in which case both the backslash and the newline are stripped.
- A single quote (') quotes everything up to the next single quote (this may span lines).
- A double quote (") quotes all characters, except $, ` and \, up to the next unquoted double quote. The $ and ` characters inside double quotes have their usual meaning (i.e., parameter, command or arithmetic substitution), except no field splitting is carried out on the results of double-quoted substitutions.

  If a \ inside a double-quoted string is followed by \, $, ` or ", it's replaced by the second character; if it's followed by a newline, both the \ and the newline are stripped; otherwise, both the \ and the character following are unchanged.

See "*POSIX mode* (p. 1056)," below, for a special rule regarding sequences of the form "…`…\"…`…".

Aliases

There are two types of aliases: normal command aliases and tracked aliases. Command aliases are normally used as shorthand for a long or often used command. The shell expands command aliases (i.e., substitutes the alias name for its value) when it reads the first word of a command. An expanded alias is reprocessed to check for more aliases. If a command alias ends in a space or tab, the following word is also checked for alias expansion. The alias expansion process stops when a word that isn't an alias is found, when a quoted word is found or when an alias word that's currently being expanded is found.

The shell automatically defines the following command aliases:

```
autoload='typeset -fu'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
local='typeset'
```


Copyright © 2014, QNX Software Systems Limited

```
login='exec login'
newgrp='exec newgrp'
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'
```

Tracked aliases allow the shell to remember where it found a particular command. The first time the shell does a path search for a command that's marked as a tracked alias, it saves the full path of the command. The next time the command is executed, the shell checks the saved path to see that it's still valid, and if so, avoids repeating the path search. You can create and list tracked aliases by using *alias -t* (p. 1060).

Changing the ***PATH*** parameter clears the saved paths for all tracked aliases.

If the trackall option is set (i.e., `set -o trackall` or `set -h`), the shell tracks all commands. This option is set automatically for noninteractive shells. For interactive shells, only the following commands are automatically tracked:

- *cat* (p. 95)
- *cc* (p. 97)
- *chmod* (p. 124)
- *cp* (p. 141)
- *date* (p. 170)
- *grep* (p. 914)
- *ls* (p. 1139)
- *make* (p. 1166)
- *mv* (p. 1328)
- *pr* (p. 1571)
- *rm* (p. 1673)
- *sed* (p. 1732)
- *sh* (p. 1760)

Substitution

The first step the shell takes in executing a simple command is to perform substitutions on the words of the command. There are three kinds of substitution:

This substitution:	Takes the form:
Parameter	$*name* or ${…} (see "*Parameters* (p. 1040)," below)
Command	$(*command*) or `` `command` ``
Arithmetic	$((*expression*))

Copyright © 2014, QNX Software Systems Limited

If a substitution appears outside of double quotes, the results of the substitution are generally subject to word or field splitting according to the current value of the ***IFS*** (internal field separators) parameter.

The ***IFS*** parameter specifies a list of characters that are used to break a string up into several words; any characters from the set space, tab and newline that appear in the IFS characters are called *IFS whitespace*. Sequences of one or more IFS whitespace characters, in combination with zero or one non-IFS whitespace characters delimit a field. As a special case, leading and trailing IFS whitespace is stripped (i.e., no leading or trailing empty field is created by it); leading or trailing non-IFS whitespace does create an empty field.

For example: if ***IFS*** is set to `<space>:`, the sequence of characters `<space>A<space>:<space><space>B::D` contains four fields: `A`, `B`, an empty field, and `D`. Note that if the ***IFS*** parameter is set to the null string, no field splitting is done; if the parameter is unset, the default value of space, tab, and newline is used.

The results of substitution are, unless otherwise specified, also subject to brace expansion and filename expansion (see the relevant sections below).

A command substitution is replaced by the output generated by the specified command, which is run in a subshell. For `$(`*command*`)` substitutions, normal quoting rules are used when *command* is parsed, however, for the `` `command` `` form, a \ followed by any of $, ` or \ is stripped (a \ followed by any other character is unchanged).

As a special case in command substitutions, a command of the form `< `*file* is interpreted to mean substitute the contents of *file*. For example, `$(< foo)` has the same effect as `$(cat foo)`, but the former is carried out more efficiently because no process is started.

`$(`*command*`)` expressions are currently parsed by finding the matching parenthesis, regardless of quoting. This will hopefully be fixed soon.

Arithmetic substitutions are replaced by the value of the specified expression. For example, the command `echo $((2+3*4))` prints 14. See "*Arithmetic expressions* (p. 1052)" for a description of an *expression*.

Parameters

Parameters are shell variables; you can assign values to them and access their values using a parameter substitution. A parameter name is either one of the special single punctuation or digit character parameters described below, or a letter followed by zero or more letters or digits (the underscore (_) counts as a letter). Parameter substitutions take the form $*name* or ${*name*}, where *name* is a parameter name. If substitution is performed on a parameter that isn't set, a null string is substituted unless the nounset option (`set -o nounset` or `set -u`) is set, in which case an error occurs.

You can assign values to parameters in a number of ways:

Copyright © 2014, QNX Software Systems Limited

- The shell implicitly sets some parameters, such as *#*, ***PWD***, and so on; this is the only way the special single character parameters are set.
- Parameters are imported from the shell's environment at startup.
- You can assign values to parameters on the command line. For example, `FOO=bar` sets the parameter *FOO* to `bar`; you can do multiple parameter assignments on a single command line and follow them by a simple command, in which case the assignments are in effect only for the duration of the command (such assignments are also exported — see below for implications of this).

You must leave both the parameter name and the = unquoted for the shell to recognize a parameter assignment.

- You can set parameters with the *export* (p. 1063), *readonly* (p. 1066), and *typeset* (p. 1077) commands; see their descriptions in "*Command execution and builtin commands* (p. 1058)."
- The `for` and `select` loops set parameters as well as the *getopts* (p. 1064), *read* (p. 1066), and *set -A* (p. 1067) commands.
- You can assign values to parameters using assignment operators inside arithmetic expressions (see "*Arithmetic expressions* (p. 1052)," below) or using the `${`*name*`=`*value*`}` form of parameter substitution (see below).

Parameters with the export attribute (set using the *export* (p. 1063) or *typeset -x* (p. 1077) commands, or by parameter assignments followed by simple commands) are put in the environment (see *environ()* in the *C Library Reference*) of commands run by the shell as *name*=*value* pairs. The order in which parameters appear in the environment of a command is unspecified. When the shell starts up, it extracts parameters and their values from its environment and automatically sets the export attribute for those parameters.

You can apply modifiers to the `${`*name*`}` form of parameter substitution:

${*name*:-*word*}

If *name* is set and not null, it's substituted, otherwise *word* is substituted.

${*name*:+*word*}

If *name* is set and not null, *word* is substituted, otherwise nothing is substituted.

${*name*:=*word*}

If *name* is set and not null, it's substituted, otherwise it's assigned *word* and the resulting value of *name* is substituted.

${*name*:?*word*}

Copyright © 2014, QNX Software Systems Limited

If *name* is set and not null, it's substituted, otherwise *word* is printed on standard error (preceded by *name*:) and an error occurs (normally causing termination of a shell script, function or .-script). If *word* is omitted, the string `parameter null or not set` is used instead.

In the above modifiers, you can omit the `:`, in which case the conditions depend only on *name*'s being set (as opposed to set and not null). If *word* is needed, parameter, command, arithmetic and tilde substitution are performed on it; if *word* isn't needed, it isn't evaluated.

You can also use the following forms of parameter substitution:

${#*name*}

The number of positional parameters if *name* is *, @ or isn't specified, or the length of the string value of parameter *name*.

${#*name*[*]} or ${#*name*[@]}

The number of elements in the array *name*.

${*name*#*pattern*} or ${*name*##*pattern*}

If *pattern* matches the beginning of the value of parameter *name*, the matched text is deleted from the result of substitution. A single # results in the shortest match, two results in the longest match.

${*name*%*pattern*} or ${*name*%%*pattern*}

Like ${...#...} substitution, but it deletes from the end of the value.

The shell implicitly set the following special parameters; you can't set them directly using assignments:

!

The process ID of the last background process started. If no background processes have been started, the parameter isn't set.

#

The number of positional parameters (i.e., *$1*, *$2*, and so on).

$

The process ID of the shell, or the PID of the original shell if it's a subshell.

-

The concatenation of the current single letter options (see the *set* (p. 1067) command below for a list of options).

Copyright © 2014, QNX Software Systems Limited

?

The exit status of the last nonasynchronous command executed. If the last command was killed by a signal, *$?* is set to 128 plus the signal number.

0

The name the shell was invoked with (i.e., *argv*[0]), or the *command-name* if it was invoked with the -c option and the *command-name* was supplied, or the *file* argument, if it was supplied. If the posix option isn't set, `$0` is the name of the current function or script.

1 ... 9

The first nine positional parameters that were supplied to the shell, function or .-script. You can access further positional parameters by using ${*number*}.

*****

All positional parameters (except parameter 0), i.e., *$1 $2 $3* and so on. If used outside of double quotes, parameters are separate words (which are subjected to word splitting); if used within double quotes, parameters are separated by the first character of the ***IFS*** parameter (or the empty string if ***IFS*** is null).

@

Same as $*, unless it's used inside double quotes, in which case a separate word is generated for each positional parameter — if there are no positional parameters, no word is generated (you can use $@ to access arguments, verbatim, without losing null arguments or splitting arguments with spaces).

The shell sets and/or uses the following parameters:

_ (underscore)

When an external command is executed by the shell, this parameter is set in the environment of the new process to the path of the executed command. In interactive use, this parameter is also set in the parent shell to the last word of the previous command. When ***MAILPATH*** messages are evaluated, this parameter contains the name of the file that changed (see ***MAILPATH*** parameter below).

CDPATH

The search path for the *cd* (p. 1061) builtin command. Works the same way as ***PATH*** for those directories not beginning with `/` in `cd` commands. Note that if ***CDPATH*** is set and doesn't contain `.` or an empty path, the current directory isn't searched.

Copyright © 2014, QNX Software Systems Limited

COLUMNS

Set to the number of columns in the terminal or window. Currently set to the *cols* value as reported by *stty* (p. 1863) if that value is nonzero. This parameter is used by the interactive line editing modes, and by `select`, *set -o* (p. 1067), and *kill -l* (p. 1065) commands to format information in columns.

EDITOR

If the ***VISUAL*** parameter isn't set, this parameter controls the command line editing mode for interactive shells. See ***VISUAL*** parameter below for how this works.

ENV

If this environment variable is found to be set after any profile files are executed, the expanded value is used as a shell start-up file. It typically contains function and alias definitions. People frequently call this file `.kshrc`, but you can give it whatever name you like.

EXECSHELL

If set, this parameter is assumed to contain the shell that's to be used to execute commands that *execve()* fails to execute and that don't start with a `#!` *shell* sequence.

FCEDIT

The editor used by the *fc* (p. 1063) command (see below).

FPATH

Like ***PATH***, but used when an undefined function is executed to locate the file defining the function. It's also searched when a command can't be found using ***PATH***. For more information, see "*Functions* (p. 1055)," below.

HISTFILE

The name of the file used to store history. When assigned to, history is loaded from the specified file. Also, several invocations of the shell running on the same machine share their history if their ***HISTFILE*** parameters all point at the same file.

If ***HISTFILE*** isn't set, no history file is used. This is different from the original Korn shell, which uses ***$HOME/***`.sh_history`; in the future, `ksh` may also use a default history file.

HISTSIZE

Copyright © 2014, QNX Software Systems Limited

The number of commands normally stored for history, default 128.

HOME

The default directory for the *cd* (p. 1061) command and the value substituted for an unqualified ~ (see "*Tilde expansion* (p. 1047)," below).

IFS

Internal field separator, used during substitution and by the *read* (p. 1066) command, to split values into distinct arguments; normally set to space, tab and newline. See "*Substitution* (p. 1039)," above, for details.

This parameter isn't imported from the environment when the shell is started.

KSH_VERSION

The version of shell and the date the version was created (readonly). See also the version commands in "*emacs interactive input-line editing* (p. 1084)" below.

MAIL

If set, you're told of the arrival of mail in the named file. This parameter is ignored if the ***MAILPATH*** parameter is set.

MAILCHECK

How often, in seconds, the shell checks for mail in the file(s) specified by ***MAIL*** or ***MAILPATH***. If 0, the shell checks before each prompt. The default is 600 (10 minutes).

MAILPATH

A list of files to be checked for mail. The list is separated by colons, and each file may be followed by a ? and a message to be printed if new mail has arrived. Command, parameter and arithmetic substitution is performed on the message, and, during substitution, the parameter *$_* contains the name of the file. The default message is `you have mail in $_`.

OLDPWD

The previous working directory. Unset if *cd* (p. 1061) hasn't successfully changed directories since the shell started, or if the shell doesn't know where it is.

OPTARG

Copyright © 2014, QNX Software Systems Limited

When using *getopts* (p. 1064), this parameter contains the argument for a parsed option, if it requires one.

OPTIND

The index of the last argument processed when using *getopts* (p. 1064). Assigning 1 to this parameter causes `getopts` to process arguments from the beginning the next time it's invoked.

PATH

A colon-separated list of directories that are searched when looking for commands and `.`'d files. An empty string resulting from a leading or trailing colon, or two adjacent colons is treated as a `.`, the current directory. For more information on setting ***PATH***, see "Setting ***PATH*** and ***LD_LIBRARY_PATH***" in the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*.

POSIXLY_CORRECT

If set, this parameter causes the posix option to be enabled. See "*POSIX mode* (p. 1056)," below.

PPID

The process ID of the shell's parent (read only).

PS1

The primary prompt for interactive shells. Parameter, command and arithmetic substitutions are performed, and `!` is replaced with the current command number (see the *fc* (p. 1063) command below). You can put a literal `!` in the prompt by placing `!!` in ***PS1***.

Note that since the command line editors try to figure out how long the prompt is (so they know how far it is to the edge of the screen), escape codes in the prompt tend to mess things up. You can tell the shell not to count certain sequences (such as escape codes) by prefixing your prompt with a nonprinting character (such as **Ctrl**–**A**) followed by a carriage return and then delimiting the escape codes with this nonprinting character. If you don't have any nonprinting characters, you're out of luck... BTW, don't blame me for this hack; it's in the original `ksh`. Default is `$` for non-`root` users, # for `root`.

PS2

Secondary prompt string, by default `"> "`, used when more input is needed to complete a command.

PS3

Copyright © 2014, QNX Software Systems Limited

Prompt used by the `select` statement when reading a menu selection. Default is `"#? "`.

PS4

Used to prefix commands that are printed during execution tracing (see the *set -x* (p. 1067) command below). Parameter, command and arithmetic substitutions are performed before it's printed. Default is `"+ "`.

PWD

The current working directory. May be unset or null if the shell doesn't know where it is.

RANDOM

A simple random number generator. Every time ***RANDOM*** is referenced, it's assigned the next number in a random number series. You can set the point in the series by assigning a number to ***RANDOM*** (see *rand()*).

REPLY

Default parameter for the *read* (p. 1066) command if no names are given. Also used in `select` loops to store the value that's read from standard input.

SECONDS

The number of seconds since the shell started or, if the parameter has been assigned an integer value, the number of seconds since the assignment plus the value that was assigned.

TMOUT

If set to a positive integer in an interactive shell, it specifies the maximum number of seconds the shell waits for input after printing the primary prompt (***PS1***). If the time is exceeded, the shell exits.

TMPDIR

The directory in which to create shell temporary files. If this parameter isn't set, or doesn't contain the absolute path of a writable directory, temporary files are created in `/tmp`.

VISUAL

If set, this parameter controls the command-line editing mode for interactive shells. If the last component of the path specified in this parameter contains the string `emacs` or `gmacs`, the `emacs` or `gmacs` (Gosling `emacs`) editing mode is enabled, respectively.

Tilde expansion

Copyright © 2014, QNX Software Systems Limited

Tilde expansion, which is done in parallel with parameter substitution, is done on words starting with an unquoted ~. The characters following the tilde, up to the first slash (/), if any, are assumed to be a login name. If the login name is empty, + or -, the value of the ***HOME***, ***PWD***, or ***OLDPWD*** parameter is substituted, respectively. Otherwise, the password file is searched for the login name, and the tilde expression is substituted with the user's home directory. If the login name isn't found in the password file or if any quoting or parameter substitution occurs in the login name, no substitution is performed.

In parameter assignments (those preceding a simple command or those occurring in the arguments of *alias* (p. 1060), *export* (p. 1063), *readonly* (p. 1066), and *typeset* (p. 1077)), tilde expansion is done after any unquoted colon (:), and login names are also delimited by colons.

The home directory of previously expanded login names are cached and reused. You can use the `alias -d` command to list, change and add directory aliases to this cache. For example:

```
alias -d fac=/usr/local/facilities; cd ~fac/bin
```

Brace expansion (alternation)

Brace expressions, which take the form:

prefix{str1,…,strN}suffix

are expanded to *N* words, each of which is the concatenation of *prefix*, *stri* and *suffix*. For example, `a{c,b{X,Y},d}e` expands to four words: `ace`, `abXe`, `abYe`, and `ade`). As noted in the example, you can nest brace expressions, and the resulting words aren't sorted. Brace expressions must contain an unquoted comma (,) for expansion to occur (i.e., `{}` and `{foo}` aren't expanded). Brace expansion is carried out after parameter substitution and before filename generation.

Filename patterns

A filename pattern is a word containing one or more unquoted ? or * characters or […] sequences. Once brace expansion has been performed, the shell replaces filename patterns with the sorted names of all the files that match the pattern (if no files match, the word is left unchanged). The pattern elements have the following meanings:

?

Matches any single character.

*****

Matches any sequence of characters.

[…]

Copyright © 2014, QNX Software Systems Limited

Matches any of the characters inside the brackets. You can specify ranges of characters by separating two characters with a `-`. For example, `[a0-9]` matches the letter `a` or any digit. In order for a `-` to represent itself, you must either quote it or use it as the first or last character in the character list. Similarly, you must quote a `]` or use it as the first character in the list if it's to represent itself instead of the end of the list.

Also, a `!` appearing at the start of the list has special meaning (see below), so to represent itself, you must quote it or use it later in the list.

`[!…]`

Like `[…]`, except it matches any character not inside the brackets.

`*(`*`pattern`*`|…|`*`pattern`*`)`

Matches any string of characters that matches zero or more occurrences of the specified patterns. For example, the pattern `*(foo|bar)` matches the empty string, as well as the strings `foo`, `bar`, `foobarfoo`, and so on.

`+(`*`pattern`*`|…|`*`pattern`*`)`

Matches any string of characters that matches one or more occurrences of the specified patterns. For example, the pattern `+(foo|bar)` matches the strings `foo`, `bar`, `foobarfoo`, and so on.

`?(`*`pattern`*`|…|`*`pattern`*`)`

Matches the empty string or a string that matches one of the specified patterns. For example, the pattern `?(foo|bar)` matches only the empty string, `foo` and `bar`.

`@(`*`pattern`*`|…|`*`pattern`*`)`

Matches a string that matches one of the specified patterns. For example, the pattern `@(foo|bar)` matches only the strings `foo` and `bar`.

`!(`*`pattern`*`|…|`*`pattern`*`)`

Matches any string that doesn't match one of the specified patterns. For example: the pattern `!(foo|bar)` matches all strings except `foo` and `bar`; the pattern `!(*)` matches no strings; the pattern `!(?)*` matches all strings (think about it).

Note that none of the above pattern elements match either a period (`.`) at the start of a filename or a slash (`/`), even if they are explicitly used in a `[…]` sequence.

Copyright © 2014, QNX Software Systems Limited

In QNX Neutrino 6.5.0 and later, the pattern `.*` matches the `.` and `..` names. This could cause problems with scripts that assume that the shell never matches these names.

If the markdirs option is set, any directories that result from filename generation are marked with a trailing `/`.

The POSIX character classes (i.e., `[:class-name:]` inside a `[...]` expression) aren't yet implemented.

Input/output redirection

When a command is executed, its standard input, standard output and standard error (file descriptors 0, 1 and 2, respectively) are normally inherited from the shell. Three exceptions to this are commands in pipelines, for which standard input and/or standard output are those set up by the pipeline, asynchronous commands created when job control is disabled, for which standard input is initially set to be from `/dev/null`, and commands for which any of the following redirections have been specified:

> file

Standard output is redirected to *file*. If *file* doesn't exist, it's created; if it does exist, is a regular file and the noclobber option is set, an error occurs, otherwise the file is truncated. Note that this means the command `cmd < foo > foo` opens `foo` for reading and then truncates it when it opens it for writing, before `cmd` gets a chance to actually read it.

>| file

Same as >, except the file is truncated, even if the noclobber option is set.

>> file

Same as >, except an existing file is appended to instead of being truncated. Also, the file is opened in append mode, so writes always go to the end of the file (see *open()*).

< file

Standard input is redirected from *file*, which is opened for reading.

<> file

Same as <, except the file is opened for reading and writing.

<< marker

Copyright © 2014, QNX Software Systems Limited

After reading the command line containing this kind of redirection (called a *here document*), the shell copies lines from the command source into a temporary file until a line matching *marker* is read. When the command is executed, standard input is redirected from the temporary file.

The line at the end of the "here document" must match *marker* exactly; it must not have any leading or trailing whitespace characters.

If *marker* contains no quoted characters, the contents of the temporary file are processed as if enclosed in double quotes each time the command is executed, so parameter, command and arithmetic substitutions are performed, along with backslash (\) escapes for `$`, `` ` ``, `\` and `\newline`. If multiple here documents are used on the same command line, they're saved in order.

<<- *marker*

Same as <<, except leading tabs are stripped from lines in the here document.

<& *fd*

Standard input is duplicated from file descriptor *fd*. The *fd* can be a single digit, indicating the number of an existing file descriptor, the letter `p`, indicating the file descriptor associated with the output of the current coprocess, or the character `-`, indicating standard input is to be closed.

>& *fd*

Same as <&, except the operation is done on standard output.

In any of the above redirections, you can explicitly give the file descriptor that's redirected (i.e., standard input or standard output) by preceding the redirection with a single digit. Parameter, command and arithmetic substitutions, tilde substitutions and filename generation are all performed on the *file*, *marker* and *fd* arguments of redirections. Note however, that the results of any filename generation are only used if a single file is matched; if multiple files match, the word with the unexpanded filename generation characters is used. Note that in restricted shells, you can't use redirections that can create files.

For simple commands, redirections may appear anywhere in the command; for compound commands (`if` statements, and so on), any redirections must appear at the end. Redirections are processed after pipelines are created and in the order they are given, so:

```
cat /foo/bar 2>&1 > /dev/null | cat -n
```

Copyright © 2014, QNX Software Systems Limited

prints an error with a line number prepended to it.

Arithmetic expressions

You can use integer arithmetic expressions with the *let* (p. 1065) command, inside `$(( ... ))` expressions, inside array references (e.g., *name*[*expr*]), as numeric arguments to the *test* (p. 1071) command, and as the value of an assignment to an integer parameter.

Expressions may contain alphanumeric parameter identifiers, array references, and integer constants. You can combine expressions with the following C operators (listed and grouped in increasing order of precedence):

Unary operators

```
+ - ! ~ ++ --
```

Binary operators

```
,
= *= /= %= += -= <<= >>= &= ^= |=
||
&&
|
^
&
== !=
< <= >= >
<< >>
+ -
* / %
```

Ternary operator

`?:` (Precedence is immediately higher than assignment.)

Grouping operators

`( )`

You can specify integer constants with arbitrary bases by using the notation *base#number*, where *base* is a decimal integer specifying the base, and *number* is a number in the specified base.

The operators are evaluated as follows:

Unary +

Result is the argument (included for completeness).

Unary -

Negation.

!

Copyright © 2014, QNX Software Systems Limited

Logical not; the result is 1 if argument is zero, 0 if not.

~

Arithmetic (bitwise) not.

++

Increment; you must apply it to a parameter (not a literal or other expression) — the parameter is incremented by 1. When used as a prefix operator, the result is the incremented value of the parameter, when used as a postfix operator, the result is the original value of the parameter.

--

Similar to ++, except the parameter is decremented by 1.

,

Separates two arithmetic expressions; the left hand side is evaluated first, then the right. The result is value of the expression on the right hand side.

=

Assignment; the variable on the left is set to the value on the right.

***= /= %= += -= <<= >>= &= ^= |=**

Assignment operators:

```
var op= expr
```

is the same as:

```
var = var op ( expr )
```

||

Logical OR; the result is 1 if either argument is nonzero, 0 if not. The right argument is evaluated only if the left argument is zero.

&&

Logical AND; the result is 1 if both arguments are nonzero, 0 if not. The right argument is evaluated only if the left argument is nonzero.

|

Arithmetic (bitwise) OR.

^

Arithmetic (bitwise) exclusive-OR.

Copyright © 2014, QNX Software Systems Limited

&

Arithmetic (bitwise) AND.

==

Equal; the result is 1 if both arguments are equal, 0 if not.

!=

Not equal; the result is 0 if both arguments are equal, 1 if not.

<

Less than; the result is 1 if the left argument is less than the right, 0 if not.

<= >= >

Less than or equal, greater than or equal, greater than. See <.

<< >>

Shift left (right); the result is the left argument with its bits shifted left (right) by the amount given in the right argument.

+ - * /

Addition, subtraction, multiplication, and division.

%

Remainder; the result is the remainder of the division of the left argument by the right. The sign of the result is unspecified if either argument is negative.

arg1 ? arg2 : arg3

If *arg1* is nonzero, the result is *arg2*, otherwise *arg3*.

Coprocesses

A coprocess, which is a pipeline created with the |& operator, is an asynchronous process that the shell can both write to (using *print -p* (p. 1065)) and read from (using *read -p* (p. 1066)). The input and output of the coprocess can also be manipulated using >&p and <&p redirections, respectively. Once a coprocess has been started, another can't be started until the coprocess exits, or until the coprocess input has been redirected using an *exec n>&p* (p. 1063) redirection. If a coprocess's input is redirected in this way, the next coprocess to be started shares the output with the first coprocess, unless the output of the initial coprocess has been redirected using an exec *n*<&p redirection.

Some notes concerning coprocesses:

Copyright © 2014, QNX Software Systems Limited

- The only way to close the coprocess input (so the coprocess reads an end-of-file) is to redirect the input to a numbered file descriptor and then close that file descriptor (e.g., `exec 3>&p;exec 3>&-`).
- In order for coprocesses to share a common output, the shell must keep the write portion of the output pipe open. This means that end of file won't be detected until all coprocesses sharing the coprocess output have exited (when they all exit, the shell closes its copy of the pipe). You can avoid this by redirecting the output to a numbered file descriptor (as this also causes the shell to close its copy).

This behavior is slightly different from the original Korn shell, which closes its copy of the write portion of the coprocess's output when the most recently started coprocess (instead of when all sharing coprocesses) exits.

- The *print -p* (p. 1065) command ignores `SIGPIPE` signals during writes if the signal isn't being trapped or ignored; the same isn't true if the coprocess input has been duplicated to another file descriptor and `print -u`*n* is used.

Functions

Functions are defined using either Korn shell `function` *name* syntax or the Bourne/POSIX shell *name*() syntax (see below for the difference between the two forms). Functions are like .-scripts in that they are executed in the current environment, however, unlike .-scripts, shell arguments (i.e., positional parameters, *$1*, and so on) are never visible inside them. When the shell is determining the location of a command, functions are searched after special builtin commands, and before regular and nonregular builtins, and before the ***PATH*** is searched.

You can delete an existing function by using *unset -f function-name* (p. 1081). You can list the functions by executing *typeset +f* (p. 1077), and list the function definitions by executing `typeset -f`. You can use the `autoload` command (which is an alias for `typeset -fu`) to create undefined functions; when an undefined function is executed, the shell searches the path specified in the ***FPATH*** parameter for a file with the same name as the function, which, if found is read and executed. If after executing the file, the named function is found to be defined, the function is executed, otherwise, the normal command search is continued (i.e., the shell searches the regular builtin command table and ***PATH***). Note that if a command isn't found using ***PATH***, the shell tries to autoload a function using ***FPATH*** (this is an undocumented feature of the original Korn shell).

Functions can have two attributes, *trace* and *export*, which you set with `typeset -ft` and `typeset -fx`. When a traced function is executed, the shell's xtrace option is turned on for the function's duration, otherwise the xtrace option is turned off. The export attribute of functions is currently not used. In the original Korn shell, exported functions are visible to shell scripts that are executed.

Copyright © 2014, QNX Software Systems Limited

Since functions are executed in the current shell environment, parameter assignments made inside functions are visible after the function completes. If this isn't the desired effect, you can use the *typeset* (p. 1077) command inside a function to create a local parameter. Note that special parameters (e.g., `$$`, `$!`) can't be scoped in this way.

The exit status of a function is that of the last command executed in the function. You can make a function finish immediately by using the *return* (p. 1067) command; you can also use this to explicitly specify the exit status.

Functions defined with the `function` reserved word are treated differently in the following ways from functions defined with the `()` notation:

- The `$0` parameter is set to the name of the function (Bourne-style functions leave `$0` untouched).
- Parameter assignments preceding function calls aren't kept in the shell environment (executing Bourne-style functions keeps assignments).
- ***OPTIND*** is saved/reset and restored on entry and exit from the function so you can use *getopts* (p. 1064) properly both inside and outside the function (Bourne-style functions leave ***OPTIND*** untouched, so using `getopts` inside a function interferes with using `getopts` outside the function).

In the future, the following differences will also be added:

- A separate trap/signal environment will be used during the execution of functions. This will mean that traps set inside a function won't affect the shell's traps and signals that aren't ignored in the shell (but may be trapped) will have their default effect in a function.
- The EXIT trap, if set in a function, will be executed after the function returns.

POSIX mode

The shell is intended to be POSIX compliant, however, in some cases, POSIX behavior is contrary either to the original Korn shell behavior or to user convenience. How the shell behaves in these cases is determined by the state of the posix option (*set -o posix* (p. 1067)); if it's on, the POSIX behavior is followed, otherwise it isn't. The posix option is set automatically when the shell starts up if the environment contains the ***POSIXLY_CORRECT*** parameter. (The shell can also be compiled so that it's in POSIX mode by default, however this usually isn't desirable).

The following is a list of things that are affected by the state of the posix option:

`\"` inside double quoted `` `...` `` command substitutions

In POSIX mode, the `\"` is interpreted when the command is interpreted; in non-POSIX mode, the backslash is stripped before the command substitution is interpreted. For example, `echo "`echo \"hi\"`"` produces `"hi"` in POSIX mode, `hi` in non-POSIX mode. To avoid problems, use the `$(...)` form of command substitution.

Copyright © 2014, QNX Software Systems Limited

kill -l **(p. 1065) output**

In POSIX mode, signal names are listed one a single line; in non-POSIX mode, signal numbers, names and descriptions are printed in columns. In future, a new option (-v perhaps) will be added to distinguish the two behaviors.

fg **(p. 1064) exit status**

In POSIX mode, the exit status is 0 if no errors occur; in non-POSIX mode, the exit status is that of the last foregrounded job.

getopts **(p. 1064)**

In POSIX mode, options must start with a `-`; in non-POSIX mode, options can start with either `-` or `+`.

Brace expansion (also known as alternation)

In POSIX mode, brace expansion is disabled; in non-POSIX mode, it's enabled. Note that *set -o posix* (p. 1067) (or setting the ***POSIXLY_CORRECT*** parameter) automatically turns the braceexpand option off, but you can explicitly turn it on later.

set - **(p. 1067)**

In POSIX mode, this doesn't clear the verbose or xtrace options; in non-POSIX mode, it does.

set **(p. 1067) exit status**

In POSIX mode, the exit status of `set` is 0 if there are no errors; in non-POSIX mode, the exit status is that of any command substitutions performed in generating the set command. For example, `set -- `false`; echo $?` prints 0 in POSIX mode, 1 in non-POSIX mode. This construct is used in most shell scripts that use the old `getopt` command.

Argument expansion of *alias* (p. 1060), *export* (p. 1063), *readonly* (p. 1066), and *typeset* (p. 1077) commands

In POSIX mode, normal argument expansion is done; in non-POSIX mode, field splitting, file globing, brace expansion and (normal) tilde expansion are turned off, and assignment tilde expansion is turned on.

Signal specification

In POSIX mode, you can specify signals as digits only if the signal numbers match POSIX values (i.e., `SIGHUP`=1, `SIGINT`=2, `SIGQUIT`=3, `SIGABRT`=6,

Copyright © 2014, QNX Software Systems Limited

SIGKILL=9, SIGALRM=14, and SIGTERM=15); in non-POSIX mode, signals can always be digits.

Alias expansion

In POSIX mode, alias expansion is only carried out when reading command words; in non-POSIX mode, alias expansion is carried out on any word following an alias that ended in a space. For example, the following `for` loop:

```
alias a='for ' i='j'
a i in 1 2; do echo i=$i j=$j; done
```

uses parameter *i* in POSIX mode, but *j* in non-POSIX mode.

Test

In POSIX mode, the expression `-t` (preceded by some number of `!` arguments) is always true, as it's a nonzero length string; in non-POSIX mode, it tests if file descriptor 1 is a tty (i.e., you can leave out the *fd* argument to the `-t` test, and it defaults to 1).

Command execution and builtin commands

After evaluation of command-line arguments, redirections and parameter assignments, the command is checked to determine its type, in this order:

1. Special builtin
2. Function
3. Regular builtin
4. Name of a file to execute found using the ***PATH*** parameter.

Special builtin commands differ from other commands in that the ***PATH*** parameter isn't used to find them, an error during their execution can cause a noninteractive shell to exit and parameter assignments that are specified before the command are kept after the command completes. Just to confuse things, if the posix option is turned off (see *set* (p. 1067) command below) some special commands are very special in that no field splitting, file globing, brace expansion nor tilde expansion is performed on arguments that look like assignments. Regular builtin commands are different only in that the ***PATH*** parameter isn't used to find them.

The original `ksh` and POSIX differ somewhat in which commands are considered special or regular.

The POSIX special commands are:

- *. (dot)* (p. 1060)
- *: (null)* (p. 1060)
- *break* (p. 1061)

Copyright © 2014, QNX Software Systems Limited

- *continue* (p. 1062)
- *eval* (p. 1062)
- *exec* (p. 1063)
- *exit* (p. 1063)
- *export* (p. 1063)
- *readonly* (p. 1066)
- *return* (p. 1067)
- *set* (p. 1067)
- *shift* (p. 1071)
- *trap* (p. 1076)
- *unset* (p. 1081)

Additional ksh special commands are:

- *builtin* (p. 1061)
- *times* (p. 1076)
- *typeset* (p. 1077)

The very special commands (non-POSIX mode) are:

- *alias* (p. 1060)
- *readonly* (p. 1066)
- *set* (p. 1067)
- *typeset* (p. 1077)

The POSIX regular commands are:

- *alias* (p. 1060)
- *bg* (p. 1061)
- *cd* (p. 1061)
- *command* (p. 1062)
- *false* (p. 1063)
- *fc* (p. 1063)
- *fg* (p. 1064)
- *getopts* (p. 1064)
- *jobs* (p. 1065)
- *kill* (p. 1065)
- *read* (p. 1066)
- *true* (p. 1076)
- *umask* (p. 1080)
- *unalias* (p. 1081)
- *wait* (p. 1081)

The additional ksh regular commands are:

Copyright © 2014, QNX Software Systems Limited

- *[* (p. 1071)
- *bind* (p. 1061)
- *echo* (p. 1062)
- *hash* (p. 1064)
- *let* (p. 1065)
- *print* (p. 1065)
- *pwd* (p. 1066)
- *test* (p. 1071)
- *ulimit* (p. 1078)
- *whence* (p. 1082)

In the future, the additional `ksh` special and regular commands may be treated differently from the POSIX special and regular commands.

Once the type of the command has been determined, any command-line parameter assignments are performed and exported for the duration of the command.

The following sections describe the builtin commands:

. (dot) builtin command

```
. file [arg …]
```

Execute the commands in *file* in the current environment. The file is searched for in the directories of ***PATH***. If arguments are given, you can use the positional parameters to access them while *file* is being executed. If no arguments are given, the positional parameters are those of the environment the command is used in.

: (null) builtin command

```
: [ … ]
```

The null command. The exit status is set to zero.

`alias` builtin command

```
alias [ -d | +-t [-r] ] [+-px] [+-] [name[=value] …]
```

Without arguments, `alias` lists all aliases. For any name without a value, the existing alias is listed. Any name with a value defines an alias (see "*Aliases* (p. 1038)," above).

When listing aliases, one of two formats is used:

- Normally, aliases are listed as *name*=*value*, where *value* is quoted.
- If options were preceded with + or a lone + is given on the command line, only *name* is printed.

In addition, if the -p option is used, each alias is prefixed with the string `alias` .

The -x option sets (+x clears) the export attribute of an alias, or, if no names are given, lists the aliases with the export attribute (exporting an alias has no affect).

Copyright © 2014, QNX Software Systems Limited

The -t option indicates that tracked aliases are to be listed/set (values specified on the command line are ignored for tracked aliases). The -r option indicates that all tracked aliases are to be reset.

The -d option causes directory aliases, which are used in tilde expansion, to be listed or set (see "*Tilde expansion* (p. 1047)," above).

bg builtin command

```
bg [job …]
```

Resume the specified stopped job(s) in the background. If no jobs are specified, %+ is assumed. This command is only available on systems that support job control. See "*Job control* (p. 1082)," below, for more information.

bind builtin command

```
bind [-m] [key[=editing-command] …]
```

Set or view the current `emacs` command editing key bindings/macros. See "*`emacs` interactive input-line editing* (p. 1084)," below, for a complete description.

break builtin command

```
break [level]
```

Exit the *level*th innermost `for`, `select`, `until`, or `while` loop. The *level* defaults to 1.

builtin builtin command

```
builtin command [arg …]
```

Execute the builtin command *command*. This is useful for explicitly executing the builtin version of commands (such as `kill`) that are also available as executable files.

cd builtin command

```
cd [-LP] [dir]
```

Set the working directory to *dir*. If the parameter ***CDPATH*** is set, it lists the search path for the directory containing *dir*. A null path means the current directory. If *dir* is missing, the home directory ***$HOME*** is used. If *dir* is `-`, the previous working directory is used (see ***OLDPWD*** parameter). If -L option (logical path) is used or if the physical option (see the *`set` (p. 1067)* command below) isn't set, references to `..` in *dir* are relative to the path used to get to the directory. If -P option (physical path) is used or if the physical option is set, `..` is relative to the filesystem directory tree. The ***PWD*** and ***OLDPWD*** parameters are updated to reflect the current and old working directory, respectively.

```
cd [-LP] old new
```

The string *new* is substituted for *old* in the current directory, and the shell attempts to change to the new directory.

Copyright © 2014, QNX Software Systems Limited

command builtin command

```
command [-pvV] cmd [arg …]
```

If neither the -v nor -V option is given, *cmd* is executed exactly as if the `command` hadn't been specified, with two exceptions:

- The *cmd* can't be a shell function.
- Special builtin commands lose their specialness (i.e., redirection and utility errors don't cause the shell to exit, and command assignments aren't permanent).

If you specify the -p option, a default search path is used instead of the current value of ***PATH***. The actual value of the default path is system-dependent: on POSIXish systems, it's the value returned by:

```
getconf _CS_PATH
```

If the -v option is given, instead of executing *cmd*, information about what would be executed is given (and the same is done for *arg* ...): for special and regular builtin commands and functions, their names are simply printed, for aliases, a command that defines them is printed, and for commands found by searching the ***PATH*** parameter, the full path of the command is printed. If no command is found, (i.e., the path search fails), nothing is printed and `command` exits with a nonzero status. The -V option is like the -v option, except it's more verbose.

continue builtin command

```
continue [level]
```

Jump to the beginning of the *level*th innermost `for`, `select`, `until`, or `while` loop. The *level* defaults to 1.

echo builtin command

```
echo [-neE] [arg …]
```

Print the arguments (separated by spaces) followed by a newline, to standard output. The newline is suppressed if any of the arguments contain the backslash sequence `\c`. See the *print* (p. 1065) command below for a list of other backslash sequences that are recognized.

The options are provided for compatibility with BSD shell scripts: -n suppresses the trailing newline, -e enables backslash interpretation (a no-op, since this is normally done), and -E suppresses backslash interpretation.

This command is also available as an executable; see *echo* (p. 668).

eval builtin command

```
eval command …
```

Concatenate the arguments (with spaces between them) to form a single string that the shell then parses and executes in the current environment.

Copyright © 2014, QNX Software Systems Limited

exec builtin command

```
exec [command [arg …]]
```

Execute the command without forking, replacing the shell process.

If no arguments are given, any IO redirection is permanent and the shell isn't replaced. Any file descriptors greater than 2 that are opened or *dup*ed in this way aren't made available to other executed commands (i.e.,. commands that aren't builtin to the shell). Note that the Bourne shell differs here: it does pass these file descriptors on.

exit builtin command

```
exit [status]
```

Exit from the shell with the specified exit status. If *status* isn't specified, the exit status is the current value of the ? parameter.

export builtin command

```
export [-p] [parameter[=value]] …
```

Set the export attribute of the named parameters. Exported parameters are passed in the environment to executed commands. If values are specified, the named parameters also assigned.

If no parameters are specified, the names of all parameters with the export attribute are printed one per line, unless the -p option is used, in which case `export` commands defining all exported parameters, including their values, are printed.

false builtin command

```
false
```

A command that exits with a nonzero status.

This command is also available as an executable; see *false* (p. 728).

fc builtin command

```
fc [-e editor | -l [-n]] [-r] [first [last]]
```

The *first* and *last* arguments select commands from the history. You can select commands by history number, or a string specifying the most recent command starting with that string. The -l option lists the command on *stdout*, and -n inhibits the default command numbers. The -r option reverses the order of the list. Without -l, the selected commands are edited by the editor specified with the -e option, or if no -e is specified, the editor specified by the ***FCEDIT*** parameter (if this parameter isn't set, `/bin/ed` is used), and then executed by the shell.

```
fc [-e - | -s] [-g] [old=new] [prefix]
```

Reexecute the selected command (the previous command by default) after performing the optional substitution of *old* with *new*. If -g is specified, all occurrences of *old* are

Copyright © 2014, QNX Software Systems Limited

replaced with *new*. This command is usually accessed with the predefined alias `r='fc -e -'`.

`fg` builtin command

```
fg [job …]
```

Resume the specified job(s) in the foreground. If no jobs are specified, %+ is assumed. This command is available only on systems that support job control. See "*Job control* (p. 1082)," below, for more information.

`getopts` builtin command

```
getopts optstring name [arg …]
```

The `getopts` command is used by shell procedures to parse the specified arguments (or positional parameters, if no arguments are given) and to check for legal options. The *optstring* argument contains the option letters that `getopts` is to recognize. If a letter is followed by a colon, the option is expected to have an argument. You can group options that don't take arguments into a single argument. If an option takes an argument and the option character isn't the last character of the argument it's found in, the remainder of the argument is taken to be the option's argument, otherwise, the next argument is the option's argument.

Each time `getopts` is invoked, it places the next option in the shell parameter *name* and the index of the next argument to be processed in the shell parameter ***OPTIND***. If the option was introduced with a +, the option placed in *name* is prefixed with a +. When an option requires an argument, `getopts` places it in the shell parameter ***OPTARG***. When an illegal option or a missing option argument is encountered, a question mark or a colon is placed in *name* (indicating an illegal option or missing argument, respectively) and ***OPTARG*** is set to the option character that caused the problem. An error message is also printed to standard error if *optstring* doesn't begin with a colon.

When the end of the options is encountered, `getopts` exits with a nonzero exit status. Options end at the first (non-option argument) argument that doesn't start with a `-`, or when a -- argument is encountered.

You can reset option parsing by setting ***OPTIND*** to 1 (this is done automatically whenever the shell or a shell procedure is invoked).

Changing the value of the shell parameter ***OPTIND*** to a value other than 1, or parsing different sets of arguments without resetting ***OPTIND*** may lead to unexpected results.

`hash` builtin command

```
hash [-r] [name …]
```


Copyright © 2014, QNX Software Systems Limited

Without arguments, any hashed executable command pathnames are listed. The -r option causes all hashed commands to be removed from the hash table. If you specify any names, the shell searches for each one as if it were a command name, and adds the name to the hash table if it's an executable command.

`jobs` builtin command

```
jobs [-lpn] [job …]
```

Display information about the specified jobs; if no jobs are specified, all jobs are displayed. The -n option causes information to be displayed only for jobs that have changed state since the last notification. If the -l option is used, the process ID of each process in a job is also listed. The -p option causes only the process group of each job to be printed. See "*Job control* (p. 1082)," below, for the format of *job* and the displayed job.

`kill` builtin command

```
kill [-s signame | -signum | -signame ] { job | pid | -pgrp } …
```

Send the specified signal to the specified jobs, process IDs, or process groups. If no signal is specified, the signal TERM is sent. If a job is specified, the signal is sent to the job's process group. See "*Job control* (p. 1082)," below, for the format of *job*.

```
kill -l [exit-status …]
```

Print the name of the signal that killed a process that exited with the specified exit-statuses. If no arguments are specified, a list of all the signals, their numbers and a short description of them are printed.

This command is also available as an executable; see *`kill`* (p. 1026).

`let` builtin command

```
let [expr …]
```

Evaluate each given expression (see "*Arithmetic expressions* (p. 1052)," above). If all expressions are successfully evaluated, the exit status is:

- 0 if the last expression evaluated to nonzero
- 1 if the last expression evaluated to zero.

If an error occurs during the parsing or evaluation of an expression, the exit status is greater than 1.

Since expressions may need to be quoted, `(( expr ))` is syntactic sugar for `let "expr"`.

`print` builtin command

```
print [-nprsun | -R [-en]] [argument …]
```

Print the arguments on the standard output, separated by spaces, and terminated with a newline. The -n option suppresses the newline. By default, certain C escapes are

Copyright © 2014, QNX Software Systems Limited

translated. These include `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, and `\0###` (*#* is an octal digit, of which there may be 0 to 3). `\c` is equivalent to using the -n option. You can inhibit backslash expansion by using the -r option. The -s option prints to the history file instead of standard output, the -u option prints to file descriptor *n* (*n* defaults to 1 if omitted), and the -p option prints to the coprocess (see "*Coprocesses* (p. 1054)," above).

The -R option is used to emulate, to some degree, the BSD echo command, which doesn't process \ sequences unless the -e option is given. As above, the -n option suppresses the trailing newline.

pwd builtin command

```
pwd [-LP]
```

Print the present working directory. If -L option is used or if the physical option (see the *set* (p. 1067) command below) isn't set, the logical path is printed (i.e., the path used to `cd` to the current directory). If -P option (physical path) is used or if the physical option is set, the path determined from the filesystem (by following `..` directories to the root directory) is printed.

This command is also available as an executable; see *pwd* (p. 1602).

read builtin command

```
read [-prsun] [parameter …]
```

Read a line of input from standard input, separate the line into fields using the ***IFS*** parameter (see "*Substitution* (p. 1039)," above), and assign each field to the specified parameters. If there are more parameters than fields, the extra parameters are set to null, or alternatively, if there are more fields than parameters, the last parameter is assigned the remaining fields (inclusive of any separating spaces). If no parameters are specified, the ***REPLY*** parameter is used. If the input-line ends in a backslash and the -r option wasn't used, the backslash and newline are stripped and more input is read. If no input is read, `read` exits with a nonzero status.

The first parameter may have a question mark and a string appended to it, in which case the string is used as a prompt (printed to standard error before any input is read) if the input is a tty (e.g., `read nfoo?'number of foos: '`).

The -u*n* and -p options cause input to be read from file descriptor *n* or the current coprocess (see "*Coprocesses* (p. 1054)," above, for comments on this), respectively. If the -s option is used, input is saved to the history file.

readonly builtin command

```
readonly [-p] [parameter[=value]] …
```

Set the readonly attribute of the named parameters. If values are given, parameters are set to them before setting the attribute. Once a parameter is made readonly, it can't be unset and its value can't be changed.

Copyright © 2014, QNX Software Systems Limited

If no parameters are specified, the names of all parameters with the readonly attribute are printed one per line, unless the -p option is used, in which case `readonly` commands defining all readonly parameters, including their values, are printed.

`return` builtin command

```
return [status]
```

Return from a function or `.` script, with exit status *status*. If no *status* is given, the exit status of the last executed command is used. If used outside of a function or `.` script, it has the same effect as *`exit`* (p. 1063). Note that `ksh` treats both profile and *$ENV* files as `.` scripts, while the original Korn shell only treats profiles as `.` scripts.

`set` builtin command

```
set [+-abCefhiklmnprsuvxX] [+-o [option]] [+-A name] [--] [arg …]
```

You can use the `set` command to set (`-`) or clear (+) shell options, set the positional parameters, or set an array parameter. You can change options by using the +-o *option* syntax, where *option* is the long name of an option, or by using the +-*letter* syntax, where *letter* is the option's single letter name (not all options have a single letter name). The following table lists both option letters (if they exist) and long names along with a description of what the option does.

Letter	Long name	Description
-A		Sets the elements of the array parameter *name* to *arg* …; if -A is used, the array is reset (i.e., emptied) first; if +A is used, the first *N* elements are set (where *N* is the number of *arg*s), the rest are left untouched.
-a	allexport	All new parameters are created with the export attribute
-b	notify	Print job notification messages asynchronously, instead of just before the prompt. Only used if job control is enabled (-m).
-C	noclobber	Prevent > redirection from overwriting existing files (you must use >| to force an overwrite).

Copyright © 2014, QNX Software Systems Limited

Letter	Long name	Description
-e	errexit	Exit (after executing the ERR trap) as soon as an error occurs or a command fails (i.e., exits with a nonzero status). This doesn't apply to commands whose exit status is explicitly tested by a shell construct such as `if`, `until`, `while`, `&&` or `||` statements.
-f	noglob	Don't expand filename patterns.
-h	trackall	Create tracked aliases for all executed commands (see "*Aliases* (p. 1038)," above). On by default for noninteractive shells.
-i	interactive	Enable interactive mode — this can only be set/unset when the shell is invoked.
-k	keyword	Parameter assignments are recognized anywhere in a command.
-l	login	The shell is a login shell — this can only be set/unset when the shell is invoked (see "*Shell startup* (p. 1030)," above).
-m	monitor	Enable job control (default for interactive shells).
-n	noexec	Don't execute any commands — useful for checking the syntax of scripts (ignored if interactive).
-p	privileged	Set automatically if, when the shell starts, the read

Copyright © 2014, QNX Software Systems Limited

Letter	Long name	Description
		uid or gid doesn't match the effective uid or gid, respectively. See "*Shell startup* (p. 1030)," above for a description of what this means.
-r	restricted	Enable restricted mode — this option can only be used when the shell is invoked. See "*Shell startup* (p. 1030)," above for a description of what this means.
-s	stdin	If used when the shell is invoked, commands are read from standard input. Set automatically if the shell is invoked with no arguments. When -s is used in the `set` command, it causes the specified arguments to be sorted before assigning them to the positional parameters (or to array *name*, if -A is used).
-u	nounset	Referencing of an unset parameter is treated as an error, unless one of the `-`, + or = modifiers is used.
-v	verbose	Write shell input to standard error as it's read.
-x	xtrace	Print commands and parameter assignments when they are executed, preceded by the value of ***PS4***.

Copyright © 2014, QNX Software Systems Limited

Letter	Long name	Description
-X	markdirs	Mark directories with a trailing / during filename generation.
	bgnice	Background jobs are run with lower priority.
	braceexpand	Enable brace expansion (also known as, alternation).
	emacs	Enable BRL `emacs`-like command line editing (interactive shells only); see "*`emacs` interactive input-line editing* (p. 1084)."
	gmacs	Enable gmacs-like (Gosling `emacs`) command line editing (interactive shells only); currently identical to `emacs` editing except that transpose (**Ctrl**–**T**) acts slightly differently.
	ignoreeof	The shell won't exit on when end-of-file is read; you have to use *`exit`* (p. 1063).
	nohup	Don't kill running jobs with a `SIGHUP` signal when a login shell exists. Currently set by default, but this will change in the future to be compatible with the original Korn shell (which doesn't have this option, but does send the `SIGHUP` signal).
	nolog	No effect — in the original Korn shell, this prevents function definitions from being stored in the history file.

Copyright © 2014, QNX Software Systems Limited

Letter	Long name	Description
	physical	Causes the *cd* (p. 1061) and *pwd* (p. 1066) commands to use physical (i.e., the filesystem's) `..` directories instead of logical directories (i.e., the shell handles `..`, which allows the user to be oblivious of symlink links to directories). Clear by default. Note that setting this option doesn't effect the current value of the ***PWD*** parameter; only the `cd` command changes ***PWD***. See the `cd` and `pwd` commands above for more details.
	posix	Enable POSIX mode. See "*POSIX mode* (p. 1056)," above.

These options can also be used upon invocation of the shell. You can find the current set of options (with single letter names) in the parameter `-`. The `set -o` command with no option name lists all the options and whether each is on or off; `set +o` prints the long names of all options that are currently on.

Remaining arguments, if any, are positional parameters and are assigned, in order, to the positional parameters (i.e., 1, 2, and so on). If options are ended with `--` and there are no remaining arguments, all positional parameters are cleared. If no options or arguments are given, then the values of all names are printed. For unknown historical reasons, a lone `-` option is treated specially: it clears both the -x and -v options.

`shift` builtin command

```
shift [number]
```

The positional parameters *number*+1, *number*+2 ... are renamed to 1, 2, and so on. The *number* defaults to 1.

`test` builtin command

```
test expression
```

Copyright © 2014, QNX Software Systems Limited

or:

```
[ expression ]
```

Evaluate the *expression* and return zero status if true, and 1 status if false and greater than 1 if there was an error. It's normally used as the condition command of `if` and `while` statements.

Calling an executable file `test` is a common mistake. If you don't specify the path to the file, the builtin command is executed.

The following basic expressions are available:

str

The *str* has nonzero length. Note that there is the potential for problems if *str* turns out to be an operator (e.g., `-r`) — it's generally better to use a test like:

```
[ X"str" != X ]
```

instead (double quotes are used in case *str* contains spaces or file globing characters).

`-r` *file*

The *file* exists and is readable.

`-w` *file*

The *file* exists and is writable.

`-x` *file*

The *file* exists and is executable.

`-a` *file*

The *file* exists.

`-e` *file*

The *file* exists.

`-f` *file*

The *file* is a regular file.

`-d` *file*

The *file* is a directory.

Copyright © 2014, QNX Software Systems Limited

-c ***file***

The *file* is a character special device.

-b ***file***

The *file* is a block special device.

-p ***file***

The *file* is a named pipe.

-u ***file***

The *file*'s mode has setuid bit set.

-g ***file***

The *file*'s mode has setgid bit set.

-k ***file***

The *file*'s mode has sticky bit set.

-s ***file***

The *file* isn't empty.

-O ***file***

The *file*'s owner is the shell's effective user-ID.

-G ***file***

The *file*'s group is the shell's effective group-ID.

-h ***file***

The *file* is a symbolic link.

-H ***file***

The *file* is a context-dependent directory (useful only on HP-UX).

-L ***file***

The *file* is a symbolic link. This is the same as `-h`.

-S ***file***

The *file* is a socket.

-o ***option***

Copyright © 2014, QNX Software Systems Limited

The shell *option* is set (see the *set* (p. 1067) command above for list of options). As a nonstandard extension, if the option starts with a `!`, the test is negated; the test always fails if option doesn't exist (thus:

```
[ -o foo -o -o !foo ]
```

returns true if and only if option *foo* exists).

file* -nt *file

The first *file* is newer than the second *file*.

file* -ot *file

The first *file* is older than the second *file*.

file* -ef *file

The first *file* is the same file as the second *file*.

-t [*fd*]

The file descriptor is a tty device. If the posix option (`set -o posix`, see "*POSIX mode* (p. 1056)," above) isn't set, you can leave out the *fd*, in which case it's taken to be 1 (the behavior differs due to the special POSIX rules described below).

string

The *string* isn't empty.

-z *string*

The *string* is empty.

-n *string*

The *string* isn't empty.

string* = *string

The strings are equal.

string* == *string

The strings are equal.

string* != *string

The strings aren't equal.

number* -eq *number

Copyright © 2014, QNX Software Systems Limited

The numbers are equal.

number* -ne *number

The numbers aren't equal.

number* -ge *number

The first number is greater than or equal to the second.

number* -gt *number

The first number is greater than the second.

number* -le *number

The first number is less than or equal to the second.

number* -lt *number

The first number is less than the second.

You can combine the above basic expressions, in which unary operators have precedence over binary operators, with the following operators (listed in increasing order of precedence):

expr* -o *expr

Logical OR.

expr* -a *expr

Logical AND.

! *expr*

Logical not.

(*expr*)

Grouping.

On operating systems not supporting `/dev/fd/`*n* devices (where *n* is a file descriptor number), the *test* (p. 1071) command attempts to fake it for all tests that operate on files (except the `-e` test). That is, `[ -w /dev/fd/2 ]` tests if file descriptor 2 is writable.

Note that some special rules are applied (courtesy of POSIX) if the number of arguments to `test` or `[ … ]` is less than five: if leading `!` arguments can be stripped such that only one argument remains, then a string length test is performed (again, even if the argument is a unary operator); if leading `!` arguments can be stripped such that three arguments remain and the second argument is a binary operator, then the binary

Copyright © 2014, QNX Software Systems Limited

operation is performed (even if first argument is a unary operator, including an unstripped `!`).

A common mistake is to use `if [ $foo = bar ]`, which fails if parameter *foo* is null or unset, if it has embedded spaces (i.e., ***IFS*** characters), or if it's a unary operator such as `!` or `-n`. Use tests like `if [ "X$foo" = Xbar ]` instead.

`times` builtin command

```
times
```

Print the accumulated user and system times used by the shell and by processes which have exited that the shell started.

`trap` builtin command

```
trap [handler signal …]
```

Set the trap handler that's to be executed when any of the specified signals are received.

The *handler* is either a null string, indicating the signals are to be ignored, a minus (`-`), indicating that the default action is to be taken for the signals (see *signal()*), or a string containing shell commands to be evaluated and executed at the first opportunity (i.e., when the current command completes, or before printing the next ***PS1*** prompt) after receipt of one of the signals.

The *signal* is the name of a signal (e.g., `SIGPIPE` or `SIGALRM`) or the number of the signal (see the *`kill -l`* (p. 1065) command above). There are two special signals:

- `EXIT` (also known as 0), which is executed when the shell is about to exit
- `ERR` which is executed after an error occurs (an error is something that causes the shell to exit if the -e or errexit option is set — see the *`set`* (p. 1067) command above).

`EXIT` handlers are executed in the environment of the last executed command. Note that for noninteractive shells, the trap handler can't be changed for signals that were ignored when the shell started.

With no arguments, `trap` lists, as a series of `trap` commands, the current state of the traps that have been set since the shell started.

The original Korn shell's `DEBUG` trap and the handling of `ERR` and `EXIT` traps in functions aren't yet implemented.

`true` builtin command

```
true
```

A command that exits with a zero value.

Copyright © 2014, QNX Software Systems Limited

This command is also available as an executable; see *true* (p. 1993).

`typeset` builtin command

```
typeset [[+-Ulprtux] [-L[n]] [-R[n]] [-Z[n]]
        [-i[n]] | -f [-tux]] [name[=value] …]
```

Display or set parameter attributes. With no *name* arguments, parameter attributes are displayed: if no options are used, the current attributes of all parameters are printed as `typeset` commands; if an option is given (or `-` with no option letter) all parameters and their values with the specified attributes are printed; if options are introduced with +, parameter values aren't printed.

If *name* arguments are given, the attributes of the named parameters are set (`-`) or cleared (+). Values for parameters may optionally be specified. If `typeset` is used inside a function, any newly created parameters are local to the function.

When -f is used, `typeset` operates on the attributes of functions. As with parameters, if no *names* are given, functions are listed with their values (i.e., definitions) unless options are introduced with +, in which case only the function names are reported.

The options are:

-L*n*

Left justify attribute: *n* specifies the field width. If *n* isn't specified, the current width of a parameter (or the width of its first assigned value) is used. Leading white space (and zeros, if used with the -Z option) is stripped. If necessary, values are either truncated or space padded to fit the field width.

-R*n*

Right justify attribute: *n* specifies the field width. If *n* isn't specified, the current width of a parameter (or the width of its first assigned value) is used. Trailing white space are stripped. If necessary, values are either stripped of leading characters or space padded to make them fit the field width.

-Z*n*

Zero fill attribute: if not combined with -L, this is the same as -R, except zero padding is used instead of space padding.

-i*n*

Integer attribute: *n* specifies the base to use when displaying the integer (if not specified, the base given in the first assignment is used). You can use arithmetic expressions to assign values to parameters that have this attribute.

-U

Copyright © 2014, QNX Software Systems Limited

Unsigned integer attribute: integers are printed as unsigned values (only useful when combined with the -i option). This option isn't in the original Korn shell.

-f

Function mode: display or set functions and their attributes, instead of parameters.

-l

Lower case attribute: all upper case characters in values are converted to lower case. (In the original Korn shell, this parameter meant "long integer" when used with the -i option).

-p

Print complete typeset commands that you can use to recreate the attributes (but not the values) of parameters. This is the default action (option exists for `ksh93` compatibility).

-r

Readonly attribute: parameters with the this attribute may not be assigned to or unset. Once this attribute is set, it can not be turned off.

-t

Tag attribute: has no meaning to the shell; provided for application use.

For functions, -t is the trace attribute. When functions with the trace attribute are executed, the xtrace (-x) shell option is temporarily turned on.

-u

Upper case attribute: all lower case characters in values are converted to upper case. (In the original Korn shell, this parameter meant "unsigned integer" when used with the -i option, which meant uppercase letters would never be used for bases greater than 10. See the -U option).

For functions, -u is the undefined attribute. See "*Functions* (p. 1055)," above for the implications of this.

-x

Export attribute: parameters (or functions) are placed in the environment of any executed commands. Exported functions aren't implemented yet.

`ulimit` builtin command

```
ulimit [-acdfHlmnpsStvw] [value]
```


Copyright © 2014, QNX Software Systems Limited

Display or set process limits. If no options are used, the file size limit (-f) is assumed. The *value*, if specified, may be either be an arithmetic expression or the word `unlimited`. The limits affect the shell and any processes created by the shell after they're imposed. Note that some systems may not allow limits to be increased once they are set. Also note that the types of limits available are system dependent — some systems have only the -f limit.

The options are:

-a

Display all limits; unless -H is used, soft limits are displayed.

-H

Set the hard limit only (default is to set both hard and soft limits).

-S

Set the soft limit only (default is to set both hard and soft limits).

-c

Impose a size limit of *n* blocks on the size of core dumps.

-d

Impose a size limit of *n* kilobytes on the size of the data area.

-f

Impose a size limit of *n* blocks on files written by the shell and its child processes (files of any size may be read).

-l

Impose a limit of *n* kilobytes on the amount of locked (wired) physical memory.

-m

Impose a limit of *n* kilobytes on the amount of physical memory used.

-n

Impose a limit of *n* file descriptors that can be open at once.

-p

Impose a limit of *n* processes that can be run by the user at any one time.

-s

Impose a size limit of *n* kilobytes on the size of the stack area.

-t

Copyright © 2014, QNX Software Systems Limited

Impose a time limit of *n* cpu seconds to be used by each process.

-v

Impose a limit of *n* kilobytes on the amount of virtual memory used; on some systems this is the maximum allowable virtual address (in bytes, not kilobytes).

-w

Impose a limit of *n* kilobytes on the amount of swap space used. As far as `ulimit` is concerned, a block is 512 bytes.

umask builtin command

```
umask [-S] [mask]
```

Display or set the file permission creation mask, or *umask*(see *umask* (p. 2005)). If the -S option is used, the mask displayed or set is symbolic, otherwise it's an octal number.

This command is also available as an executable; see *umask* (p. 2005).

Symbolic masks are like those used by *chmod* (p. 124):

```
[ugoa]{{=+-}{rwx}*}+[,…]
```

in which the first group of characters is the *who* part, the second group is the *op* part, and the last group is the *perm* part. The *who* part specifies which part of the umask is to be modified. The letters mean:

u

The user permissions.

g

The group permissions.

o

The other permissions (nonuser, nongroup).

a

All permissions (user, group and other).

The *op* part indicates how the *who* permissions are to be modified:

=

Set.

+

Added to.

Copyright © 2014, QNX Software Systems Limited

-

Removed from.

The *perm* part specifies which permissions are to be set, added or removed:

r

Read permission.

w

Write permission.

x

Execute permission.

When symbolic masks are used, they describe what permissions may be made available (as opposed to octal masks in which a set bit means the corresponding bit is to be cleared). For example, `ug=rwx,o=` sets the mask so files won't be readable, writable or executable by "others", and is equivalent (on most systems) to the octal mask `07`.

`unalias` builtin command

```
unalias [-adt] [name …]
```

Remove the aliases for the given names. If the -a option is used, all aliases are removed. If the -t or -d option is used, the indicated operations are carried out on tracked or directory aliases, respectively.

`unset` builtin command

```
unset [-fv] parameter …
```

Unset the named parameters (-v, the default) or functions (-f). The exit status is nonzero if any of the parameters were already unset, zero otherwise.

`wait` builtin command

```
wait [job]
```

Wait for the specified job(s) to finish. The exit status of `wait` is that of the last specified job: if the last job is killed by a signal, the exit status is 128 + the number of the signal (see *`kill -l` exit-status* (p. 1065) above); if the last specified job can't be found (because it never existed, or had already finished), the exit status of wait is 127. See "*Job control* (p. 1082)," below, for the format of *job*. The `wait` command returns if a signal for which a trap has been set is received, or if a `SIGHUP`, `SIGINT` or `SIGQUIT` signal is received.

If no jobs are specified, `wait` waits for all currently running jobs (if any) to finish and exits with a zero status. If job monitoring is enabled, the completion status of jobs is printed (this isn't the case when jobs are explicitly specified).

Copyright © 2014, QNX Software Systems Limited

whence builtin command

```
whence [-pv] [name …]
```

For each name, the type of command is listed (reserved word, builtin, alias, function, tracked alias or executable). If the -p option is used, a path search is done even if *name* is a reserved word, alias, and so on. Without the -v option, `whence` is similar to *`command -v`* (p. 1062) except that `whence` finds reserved words and doesn't print aliases as alias commands; with the -v option, `whence` is the same as `command -V`. Note that for `whence`, the -p option doesn't affect the search path used, as it does for `command`. If the type of one or more of the names couldn't be determined, the exit status is nonzero.

Job control

Job control refers to the shell's ability to monitor and control *jobs*, which are processes or groups of processes created for commands or pipelines. At a minimum, the shell keeps track of the status of the background (i.e., asynchronous) jobs that currently exist; you can display this information by using the *`jobs`* (p. 1065) command. If job control is fully enabled (using *`set -m`* (p. 1067) or `set -o monitor`), as it is for interactive shells, the processes of a job are placed in their own process group, you can stop foreground jobs by typing the suspend character from the terminal (normally **Ctrl**–**Z**), you can restart jobs in either the foreground or background, using the *`fg`* (p. 1064) and *`bg`* (p. 1061) commands, and the state of the terminal is saved or restored when a foreground job is stopped or restarted.

Note that you can stop only commands that create processes (e.g., asynchronous commands, subshell commands, and nonbuiltin, nonfunction commands); you can't stop commands such as *`read`* (p. 1066).

When a job is created, it's assigned a job number. For interactive shells, this number is printed inside `[…]`, followed by the process IDs of the processes in the job when an asynchronous command is run. You can refer to a job in the `bg`, `fg`, `jobs`, *`kill`* (p. 1065) and *`wait`* (p. 1081) commands either by the process ID of the last process in the command pipeline (as stored in the *$!* parameter) or by prefixing the job number with a percent sign (%). You can also use other percent sequences to refer to jobs:

%+

The most recently stopped job, or, if there are no stopped jobs, the oldest running job.

%%, %

Same as %.

%-

The job that would be the %+ job, if the latter didn't exist.

Copyright © 2014, QNX Software Systems Limited

%n

The job with job number *n*.

%?string

The job containing the string *string* (an error occurs if multiple jobs are matched).

%string

The job starting with string *string* (an error occurs if multiple jobs are matched).

When a job changes state (e.g., a background job finishes or foreground job is stopped), the shell prints the following status information:

```
[number] flag status command
```

The fields are:

number

The job number of the job.

flag

+ or - if the job is the %+ or %- job, respectively, or space if it's neither.

status

The current state of the job:

Running

The job has neither stopped or exited (note that running doesn't necessarily mean consuming CPU time — the process could be blocked waiting for some event).

Done [(*number*)]

The job exited. The *number* is the exit status of the job, which is omitted if the status is zero.

Stopped [(*signal*)]

The job was stopped by the indicated *signal* (if no signal is given, the job was stopped by SIGTSTP).

signal-description **[(core dumped)]**

Copyright © 2014, QNX Software Systems Limited

The job was killed by a signal (e.g., Memory fault, Hangup, and so on — use *kill -l* (p. 1065) for a list of signal descriptions). The `(core dumped)` message indicates the process created a core file.

command

The command that created the process. If there are multiple processes in the job, then each process has a line showing its *command* and possibly its *status*, if it's different from the status of the previous process.

When an attempt is made to exit the shell while there are jobs in the stopped state, the shell warns the user that there are stopped jobs and doesn't exit. If another attempt is immediately made to exit the shell, the stopped jobs are sent a `SIGHUP` signal and the shell exits. Similarly, if the nohup option isn't set and there are running jobs when an attempt is made to exit a login shell, the shell warns the user and doesn't exit. If another attempt is immediately made to exit the shell, the running jobs are sent a `SIGHUP` signal and the shell exits.

`emacs` interactive input-line editing

When the emacs option is set, interactive input-line editing is enabled.

This mode is slightly different from the `emacs` mode in the original Korn shell; the 8th bit is stripped in `emacs` mode.

In this mode, various editing commands (typically bound to one or more control characters) cause immediate actions without waiting for a new-line. Several editing commands are bound to particular control characters when the shell is invoked; you can use the following commands to change these bindings:

bind **(p. 1061)**

The current bindings are listed.

`bind` ***string*=[*editing-command*]**

Bind the specified editing command to the given *string*, which should consist of a control character (which you can write using caret notation ^*X*), optionally preceded by one of the two prefix characters.

After invoking the `bind` command, typing the *string* causes the editing command to be immediately invoked. Note that although only two prefix characters (usually **ESC** and **Ctrl**–**X**) are supported, some multicharacter sequences can be supported. The following binds the arrow keys on an ANSI

Copyright © 2014, QNX Software Systems Limited

terminal, or xterm (these are in the default bindings). Of course some escape sequences won't work out quite this nicely:

```
bind '^[['=prefix-2
bind '^XA'=up-history
bind '^XB'=down-history
bind '^XC'=forward-char
bind '^XD'=backward-char
```

bind -l

List the names of the functions to which keys may be bound.

bind -m *string*=[*substitute*]

The specified input *string* is afterward immediately replaced by the given *substitute* string, which may contain editing commands.

The editing commands are listed below. Each description starts with the name of the command, a *n* (if you can prefix the command with a count), and any keys the command is bound to by default (written using caret notation, e.g., ASCII ESC character is written as `^[`). You can enter a count prefix for a command by using the sequence `^[`*n*, where *n* is a sequence of 1 or more digits; unless otherwise specified, if a count is omitted, it defaults to 1. Note that editing command names are used only with the `bind` command. Furthermore, many editing commands are useful only on terminals with a visible cursor. The default bindings were chosen to resemble corresponding `emacs` key bindings. The user's tty characters (e.g., ERASE) are bound to reasonable substitutes and override the default bindings.

abort

Key binding: `^G`

Useful as a response to a request for a `search-history` pattern in order to abort the search.

auto-insert *n*

Key binding: none

Simply causes the character to appear as literal input. Most ordinary characters are bound to this.

backward-char *n*

Key binding: `^B`

Moves the cursor backward *n* characters.

backward-word *n*

Copyright © 2014, QNX Software Systems Limited

Key binding: ^[B

Moves the cursor backward to the beginning of a word; words consist of alphanumerics, underscore (_) and dollar ($).

beginning-of-history

Key binding: ^[<

Moves to the beginning of the history.

beginning-of-line

Key binding: ^A

Moves the cursor to the beginning of the edited input line.

capitalize-word ***n***

Key binding: ^[c or ^[C

Uppercase the first character in the next *n* words, leaving the cursor past the end of the last word.

If the current line doesn't begin with a comment character, one is added at the beginning of the line and the line is entered (as if return had been pressed), otherwise the existing comment characters are removed and the cursor is placed at the beginning of the line.

complete

Key binding: ^[^[or ^I

Automatically completes as much as is unique of the command name or the filename containing the cursor. If the entire remaining command or filename is unique, a space is printed after its completion, unless it's a directory name, in which case / is appended. If there is no command or filename with the current partial word as its prefix, a bell character is output (usually causing a audio beep).

complete-command

Key binding: ^X^[

Automatically completes as much as is unique of the command name having the partial word up to the cursor as its prefix, as in the `complete` command described above.

complete-file

Key binding: ^[^X

Copyright © 2014, QNX Software Systems Limited

Automatically completes as much as is unique of the filename having the partial word up to the cursor as its prefix, as in the `complete` command described above.

`complete-list`

Key binding: `^[=`

List the possible completions for the current word.

`delete-char-backward` ***n***

Key binding: ERASE, `^?`, `^H`

Deletes *n* characters before the cursor.

`delete-char-forward` ***n***

Key binding: none

Deletes *n* characters after the cursor.

`delete-word-backward` ***n***

Key binding: `^[`ERASE, `^[^?`, `^[^H`, `^[h`

Deletes *n* words before the cursor.

`delete-word-forward` ***n***

Key binding: `^[d`

Deletes characters after the cursor up to the end of *n* words.

`down-history` ***n***

Key binding: `^N`

Scrolls the history buffer forward *n* lines (later). Each input line originally starts just after the last entry in the history buffer, so `down-history` isn't useful until either `search-history` or `up-history` has been performed.

`downcase-word` ***n***

Key binding: `^[L`, `^[l`

Lowercases the next *n* words.

`end-of-history`

Key binding: `^[>`

Moves to the end of the history.

Copyright © 2014, QNX Software Systems Limited

end-of-line

Key binding: ^E

Moves the cursor to the end of the input line.

eot

Key binding: ^_

Acts as an end-of-file; this is useful because edit-mode input disables normal terminal input canonicalization.

eot-or-delete *n*

Key binding: ^D

Acts as eot if alone on a line; otherwise acts as `delete-char-forward`.

error

Key binding: none

Error (ring the bell).

exchange-point-and-mark

Key binding: ^X^X

Places the cursor where the mark is, and sets the mark to where the cursor was.

expand-file

Key binding: ^[*

Appends a * to the current word and replaces the word with the result of performing file globbing on the word. If no files match the pattern, the bell is rung.

forward-char *n*

Key binding: ^F

Moves the cursor forward *n* characters.

forward-word *n*

Key binding: ^[f

Moves the cursor forward to the end of the *n*th word.

goto-history *n*

Copyright © 2014, QNX Software Systems Limited

Key binding: ^[g

Goes to history number *n*.

kill-line

Key binding: KILL

Deletes the entire input line.

kill-region

Key binding: ^W

Deletes the input between the cursor and the mark.

kill-to-eol *n*

Key binding: ^K

Deletes the input from the cursor to the end of the line if *n* is not specified, otherwise deletes characters between the cursor and column *n*.

list

Key binding: ^[?

Prints a sorted, columnated list of command names or filenames (if any) that can complete the partial word containing the cursor. Directory names have / appended to them.

list-command

Key binding: ^X?

Prints a sorted, columnated list of command names (if any) that can complete the partial word containing the cursor.

list-file

Key binding: ^X^Y

Prints a sorted, columnated list of filenames (if any) that can complete the partial word containing the cursor. File type indicators are appended as described under list above.

newline

Key binding: ^J, ^M

Causes the current input line to be processed by the shell. The current cursor position may be anywhere on the line.

Copyright © 2014, QNX Software Systems Limited

newline-and-next

Key binding: ^O

Causes the current input line to be processed by the shell, and the next line from history becomes the current line. This is useful only after an `up-history` or `search-history`.

no-op

Key binding: `QUIT`

This does nothing.

prefix-1

Key binding: ^[

Introduces a 2-character command sequence.

prefix-2

Key binding: ^X, ^[[

Introduces a 2-character command sequence.

prev-hist-word *n*

Key binding: ^[., ^[_

The last (*n*th) word of the previous command is inserted at the cursor.

quote

Key binding: ^^

The following character is taken literally rather than as an editing command.

redraw

Key binding: ^L

Reprints the prompt string and the current input line.

search-character-backward *n*

Key binding: ^[^]

Search backward in the current line for the *n*th occurrence of the next character typed.

search-character-forward *n*

Key binding: ^]

Copyright © 2014, QNX Software Systems Limited

Search forward in the current line for the *n*th occurrence of the next character typed.

search-history

Key binding: ^R

Enter incremental search mode. The internal history list is searched backwards for commands matching the input. An initial ^ in the search string anchors the search. The abort key leaves search mode. Other commands are executed after leaving search mode. Successive `search-history` commands continue searching backward to the next previous occurrence of the pattern. The history buffer retains only a finite number of lines; the oldest are discarded as necessary.

set-mark-command

Key binding: ^[**Space**

Set the mark at the cursor position.

stuff

Key binding: none

On systems supporting it, pushes the bound character back onto the terminal input where it may receive special processing by the terminal handler. This is useful for the BRL ^T mini-systat feature, for example.

stuff-reset

Key binding: none

Acts like `stuff`, then aborts input the same as an interrupt.

transpose-chars

Key binding: ^T

If at the end of line, or if the gmacs option is set, this exchanges the two previous characters; otherwise, it exchanges the previous and current characters and moves the cursor one character to the right.

up-history *n*

Key binding: ^P

Scrolls the history buffer backward *n* lines (earlier).

upcase-word *n*

Key binding: ^[U, ^[u

Copyright © 2014, QNX Software Systems Limited

Uppercases the next *n* words.

version

Key binding: ^V

Display the version of ksh. The current edit buffer is restored as soon as any key is pressed (the key is then processed, unless it's a space).

yank

Key binding: ^Y

Inserts the most recently killed text string at the current cursor position.

yank-pop

Key binding: ^[y

Immediately after a yank, replaces the inserted text string with the next previous killed text string.

See also:

- *The Korn Shell Command and Programming Language*, Morris Bolsky and David Korn, 1989, ISBN 0-13-516972-0.
- *UNIX Shell Programming*, Stephen G. Kochan, Patrick H. Wood, Hayden.
- *IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities*, IEEE Inc, 1993, ISBN 1-55937-255-9.

Files:

- ~/.profile
- /etc/profile
- /etc/suid_profile

Contributing author:

This shell is based on the public domain 7th edition Bourne shell clone by Charles Forsyth and parts of the BRL shell by Doug A. Gwyn, Doug Kingston, Ron Natalie, Arnold Robbins, Lou Salkind and others. The first release of pdksh was created by Eric Gisin, and it was subsequently maintained by John R. MacMillan (chance!john@sq.sq.com), and Simon J. Gerraty (sjg@zen.void.oz.au). The current maintainer is Michael Rendell (michael@cs.mun.ca). The CONTRIBUTORS file in the source distribution contains a more complete list of people and their part in the shell's development.

Report any bugs in ksh to pdksh@cs.mun.ca. Please include the version of ksh (echo $KSH_VERSION shows it), the machine, operating system and compiler you

Copyright © 2014, QNX Software Systems Limited

are using and a description of how to repeat the bug (a small shell script that demonstrates the bug is best). The following, if relevant (if you aren't sure, include them), can also helpful: options you are using (both options.h options and `set -o` options) and a copy of your `config.h` (the file generated by the configure script). You can get new versions of `ksh` from `ftp.cs.mun.ca:pub/pdksh/`.

Copyright © 2014, QNX Software Systems Limited

Chapter 13
L

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`lsm-ipfilter-v4.so`, `lsm-ipfilter-v6.so`	*`lsm-pf-v4.so`*, *`lsm-pf-v6.so`* (p. 1148)
`lsm-sctp.so`	No replacement

This chapter describes the utilities, etc. whose names start with “L”.

Copyright © 2014, QNX Software Systems Limited

ld

Linker command (POSIX)

You should use *qcc* (p. 1608) instead of calling `ld` directly.

Syntax:

ld_variant [*option*…] *objfile*

where *ld_variant* depends on the target platform, as follows:

Target platform	*ld_variant*
ARMv7	`ntoarmv7-ld`
x86	`ntox86-ld`

Runs on:

Linux, Microsoft Windows

Description:

The `ld` linker combines a number of object and archive files, relocates their data, and ties up symbol references. Usually the last step in compiling a program is to run `ld`. For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

ldd

List the shared objects that a program requires (Unix)

Syntax:

```
ldd program ...
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `ldd` ("list dynamic dependencies") command lists the shared objects that the specified programs require. If you don't specify the full path for a program, `ldd` looks for it in your current directory.

You can use this utility to determine which shared objects to include in an OS image; see the Sample Buildfiles appendix in *Building Embedded Systems*.

It's worth noting that `ldd` is also the name of the runtime linker. In the `.interp` section, it's called `ldqnx.so`, and it's in `libc`, which is why you need to create a symbolic link like this:

```
[type=link] /usr/lib/ldqnx.so.2=/proc/boot/libc.so
```

in your *mkifs* (p. 1241) buildfiles.

Examples:

```
$ ldd `which ksh`
/bin/ksh:
        libc.so.3 => /usr/lib/ldqnx.so.2 (0xb0300000)
$ ldd `which gdb`
/usr/qnx650/host/qnx6/x86/usr/bin/gdb:
        libsocket.so.2 => /lib/libsocket.so.2 (0xb8200000)
        libm.so.2 => /lib/libm.so.2 (0xb822f000)
        libc.so.3 => /usr/lib/ldqnx.so.2 (0xb0300000)
```

Environment variables:

DL_DEBUG

If this environment variable is set, the shared library loader displays debugging information about the libraries as they're opened.

Copyright © 2014, QNX Software Systems Limited

ldrel

Relocate an executable

Syntax:

```
ldrel [options] infile outfile
```

Runs on:

QNX Neutrino, Linux, Windows

Options:

-a *hex_align*

Change segment alignments (default to original alignment).

-b *hex_addr*

Program base address (default to page aligned base address).

-d *hex_addr*

Data segment address (default to just after text segment).

-f *name*

Filename holding debug information (default to *infile*).

-L

The specified stacksize can be lazy. The default is non-lazy.

-l

("el") Output LOAD segments only.

-o *hex_off*

Assume this file will be copied to another file at this offset.

-p

Pad segments so they don't share address file data.

-r

Keep relocation information in target.

-S *stacksize* [K|M]

Copyright © 2014, QNX Software Systems Limited

Note the maximum stack size in bytes, kilobytes (suffix K), or megabytes (suffix M) as a suggestion to the loader. Specifying -S 0 resets the note; the loader will use its stack size.

-s *pattern*=*filename* or -s [!]* or -s *pattern*[*]

Copy sections matching *pattern*, from the file (if given). The section must not overlay segments.

-t *hex_addr*

Text segment address (default to just after headers).

-v

Be verbose.

-x

Expand segments with zeros to make the file size equal to memory size.

-Z

Load the whole input file into memory.

Description:

The `ldrel` utility relocates executables.

Copyright © 2014, QNX Software Systems Limited

less

Display files on a page-by-page basis (UNIX)

Syntax:

```
less [-[+]aBcCdeEfimMnNqQrsSuUw] [-b n] [-x n]
     [-[z] n] [-h n] [-j n] [-p pattern]
     [-y n] [-[oO] logfile] [-t tag]
     [-T tagsfile] [+ cmd] [file...]
```

Runs on:

QNX Neutrino

Options:

Most options may be changed while `less` is running, via the dash (`-`) command.

Options are also taken from the ***LESS*** environment variable. The environment variable is parsed before the command line, so command-line options override the ***LESS*** environment variable. If an option appears in the ***LESS*** environment variable, you can reset it to its default on the command line by using the two-character combination `-+` at the beginning of the command line.

A dollar sign ($) may be used to signal the end of an option string. This is important only for options such as -t that take a following string.

-?

Display a summary of the commands accepted by `less` (the same as the `h` command). If this option is given, all other options are ignored, and `less` exits after the help screen is viewed. (Depending on how your shell interprets the question mark, it may be necessary to quote the question mark, as follows: -\?)

-a

Start searches after the last line displayed on the screen, thus skipping all lines displayed on the screen. By default, searches begin at the second line on the screen (or after the last found line; see the -j option).

-B

Disable automatic allocation of buffers, so that only the default number of buffers is used. If more data is read than fits in the buffers, the oldest data is discarded. By default, when data is coming from standard input, buffers are allocated automatically as needed to avoid loss of data.

Copyright © 2014, QNX Software Systems Limited

-b *n*

Use a nonstandard number of buffers. Buffers are 1 KB, and by default 10 buffers are used (except if data is coming from standard input; see the -B option). The number *n* specifies the number of buffers to use.

-C

Clear the screen, then do fullscreen redraws from the top line down. By default, fullscreen redraws are done by scrolling from the bottom of the screen.

-c

Do fullscreen redraws from the top line down. By default, fullscreen redraws are done by scrolling from the bottom of the screen.

-d

Suppress the error message normally displayed if the terminal is dumb (i.e., lacks some important capability, such as the ability to clear the screen or scroll backward). The -d option doesn't otherwise change the behavior of `less` on a dumb terminal.

-E

Automatically exit the first time end-of-file is reached. By default, the only way to exit `less` is via the `q` command.

-e

Automatically exit the second time end-of-file is reached. By default, the only way to exit `less` is via the `q` command.

-f

Force nonregular files (directories or device special files) to be opened. Also, suppress the warning message when a binary file is opened. By default, `less` refuses to open nonregular files.

-h *n*

Don't scroll backward any more than *n* lines. If it's necessary to scroll backward more than *n* lines, the screen is redrawn in a forward direction instead. (If the terminal can't scroll backward, -h 0 is implied.)

-i

Ignore case; uppercase and lowercase are considered identical. Also, you can search for text that's overstruck or underlined. This option is ignored if any uppercase letters appear in the search pattern.

Copyright © 2014, QNX Software Systems Limited

-j *n*

Use this line on the screen to position "target" lines. Target lines are the object of text searches, tag searches, jumps to a line number, jumps to a file percentage, and jumps to a marked position.

The screen line is specified by a number: the top line on the screen is 1, the next is 2, and so on. The number may be negative to specify a line relative to the bottom of the screen: the bottom line on the screen is -1 (the number one), the second to the bottom is -2, and so on.

With the -j option, searches begin at the line immediately after the target line. For example, if -j4 is used, the target line is the fourth line on the screen, so searches begin at the fifth line on the screen.

-M

Prompt even more verbosely than *more* (p. 1309).

-m

Prompt verbosely (like `more`), with the percent into the file. By default, `less` prompts with a colon.

-N

Display a line number at the beginning of each line.

-n

Suppress line numbers. The default (to use line numbers) may cause `less` to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with -n avoids this problem. Using line numbers means that the line number is displayed in the verbose prompt and in the = command, and the `v` command passes the current line number to the editor.

-O *file*

Copy input to the named file as it's being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, `less` *doesn't* ask for confirmation before overwriting it.

If no log file has been specified, you can use the -o and -O options from within `less` to specify a log file. Without a filename, they simply report the name of the log file.

-o *file*

Copyright © 2014, QNX Software Systems Limited

Copy input to the named file as it's being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, `less` asks for confirmation before overwriting it.

-p *pattern*

Start at the first occurrence of *pattern* in the file. The -p option on the command line is equivalent to specifying +/*pattern*.

-Q

Be totally quiet; never ring the terminal bell.

-q

Be moderately quiet; don't ring the terminal bell if an attempt is made to scroll past the end of the file or before the beginning of the file. If the terminal has a "visual bell," it's used instead. The bell is rung on certain other errors, such as typing an invalid character. The default is to ring the terminal bell in all such cases.

-r

Display "raw" control characters. The default is to display control characters using the caret (^) notation. For example, a `Ctrl-A` (001 octal) is displayed as ^A.

Note that when -r is used, `less` can't keep track of the actual appearance of the screen (since this depends on how the screen responds to each type of control character). Thus, various display problems may result, such as long lines being split in the wrong place.

-S

Chop, rather than fold, lines longer than the screen width. That is, the remainder of a long line is simply discarded. The default is to fold long lines; that is, display the remainder on the next line.

-s

Squeeze consecutive blank lines into a single blank line.

-T *tagsfile*

Use the specified tags file in place of the `tags` file.

-t *tag*

Edit the file containing the specified tag. For this to work, there must be a file called `tags` in the current directory.

Copyright © 2014, QNX Software Systems Limited

-U

Treat backspaces and carriage returns as control characters (i.e., they're handled as specified by the -r option).

By default, if neither -u nor -U is given, backspaces that appear adjacent to an underscore character are treated specially: the underlined text is displayed using the terminal's hardware underlining capability. Backspaces that appear between two identical characters are also treated specially: the overstruck text is printed using the terminal's hardware boldface capability. Other backspaces are deleted, along with the preceding character. Carriage returns immediately followed by a newline are deleted; other carriage returns are handled as specified by the -r option.

-u

Treat backspaces and carriage returns as printable characters (i.e., they're sent to the terminal when they appear in the input).

-w

Use blank lines to represent lines past the end of the file. By default, a tilde (~) character is used.

-x *n*

Set tab stops every *n* positions (the default is 8).

-y *n*

The maximum number of lines to scroll forward. If it's necessary to scroll forward more than *n* lines, the screen is repainted instead. The -c or -C option may be used to repaint from the top of the screen if desired. By default, any forward movement causes scrolling.

-[z] *n*

Change the default scrolling window size to *n* lines. The default is one screenfull. The z and w commands can also be used to change the window size. Note that the z may be omitted (as in - *n*).

+

If a command-line option begins with + (plus), the remainder of that option is taken to be an initial command to `less`. For example, +G tells `less` to start at the end of the file rather than the beginning; and +/xyz tells it to start at the first occurrence of `xyz` in the file.

As a special case, +*number* acts like +*number*g; that is, it starts the display at the specified line number (note, however, that this may be slow — see

Copyright © 2014, QNX Software Systems Limited

the g command. If the option starts with two plus signs (++), the initial command applies to every file being viewed, not just the first one. The + command (described in the "Commands" section) may also be used to set or change an initial command for every file.

file

A pathname of an input file. If no *file* operands are specified, `less` uses the standard input. If a *file* operand is the dash character (-), the standard input is read at that point of the sequence.

Description:

The `less` utility displays file on a page-by-page basis. It's similar to *more* (p. 1309), but `less` allows backward movement in the file as well as forward movement. The `less` utility uses the system terminal capability database, so it can run on a variety of terminals. There's limited support for hardcopy terminals (on a hardcopy terminal, lines that should be printed at the top of the screen are prefixed with `up-arrow`).

The `less` utility displays a screenfull of information, then prompts for user input by displaying a colon (`:`) prompt at the bottom of the screen. Commands can then be entered from the keyboard.

Commands:

Commands may be preceded by a decimal number, called *n* in the following descriptions. The number is used by some commands, as indicated.

[*n*] h

Help: display a summary of these commands. If you forget all the other commands, remember this one.

[*n*] Space ***or*** **[*n*] f**

Scroll forward *n* lines; default is one window (see the -z option). If *n* is more than the screen size, only the final screenfull is displayed.

[*n*] z

Like **Space**, but if *n* is specified, it becomes the new window size.

[*n*] Enter

Scroll forward *n* lines; the default is 1. The entire *n* lines are displayed, even if *n* is more than the screen size.

[*n*] d

Copyright © 2014, QNX Software Systems Limited

Scroll forward (down) *n* lines; the default is 1/2 screen size. If *n* is specified, it becomes the new default for subsequent d and u commands.

[*n*] b

Scroll backward *n* lines; the default is one window (see the -z option). If *n* is more than the screen size, only the final screenfull is displayed.

[*n*] w

Like b, but if *n* is specified, it becomes the new window size.

[*n*] k

Scroll backward *n* lines; the default is 1. The entire *n* lines are displayed, even if *n* is more than the screen size.

[*n*] u

Scroll backward (up) *n* lines; the default is 1/2 screen size. If *n* is specified, it becomes the new default for subsequent d and u commands.

[*n*] r

Redraw the screen.

[*n*] F

Scroll forward, and keep trying to read when the end of the file is reached. Normally this command is used when already at the end of the file. It's a way to monitor the tail of a file which is growing while it's being viewed. (The behavior is similar to the `tail -f` command.)

[*n*] g

Go to line *n* in the file; the default is 1 (beginning of file). (Note that this may be slow if *n* is large.)

[*n*] G

Go to line *n* in the file; the default is the end of the file. Note that this may be slow if *n* is large, or if *n* isn't specified and standard input, rather than a file, is being read.

[*n*] p

Go to a position *n* percent into the file. The *n* argument should be between 0 and 100. This works if standard input is being read, but only if `less` has already read to the end of the file. This is always fast, but not always useful.

[*n*] {

Copyright © 2014, QNX Software Systems Limited

If a left brace ({) appears in the top line displayed on the screen, the { command goes to the matching right brace, which is positioned on the bottom line of the screen. If there's more than one left brace on the top line, a number *n* may be used to specify the *n*th brace on the line.

[*n*] }

If a right brace (}) appears in the bottom line displayed on the screen, the } command goes to the matching left brace, which is positioned on the top line of the screen. If there's more than one right brace on the top line, a number *n* may be used to specify the *n*th brace on the line.

[*n*] (

Like {, but applies to parentheses rather than braces.

[*n*])

Like }, but applies to parentheses rather than braces.

[*n*] [

Like {, but applies to square brackets rather than braces.

[*n*]]

Like }, but applies to square brackets rather than braces.

Esc Ctrl–F *charchar*

Acts like {, but uses the two characters as open and close brackets, respectively. For example, ESC ^F <> could be used to go forward to the > that matches the < in the top displayed line.

Esc Ctrl–B *charchar*

Acts like }, but uses the two characters as open and close brackets, respectively. For example, ESC ^B <> could be used to go backward to the < that matches the > in the bottom displayed line.

m*char*

Followed by any lowercase letter, marks the current position with that letter.

' *char*

(Single quote) Followed by any lowercase letter, returns to the position that was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. Followed by a ^ or $, jumps to the beginning or end of the file

Copyright © 2014, QNX Software Systems Limited

respectively. Marks are preserved when a new file is examined, so you can use the ' command to switch between input files.

[*n*] */pattern*

Search forward in the file for the *n*th line containing the pattern; *n* defaults to 1. The pattern is a regular expression, as recognized by *sed* (p. 1732). The search starts at the second line displayed (but see the -a and -j options, which change this).

Certain characters are special if entered at the beginning of the pattern for the / and ? commands; they modify the type of search rather than become part of the pattern:

!

Search for lines that *don't* match the pattern.

*****

Search multiple files. That is, if the search reaches the end of the current file without finding a match, the search continues in the next file in the command-line list.

@

Begin the search at the first line of the first file in the command-line list, regardless of what is currently displayed on the screen or the settings of the -a or -j options.

[*n*] ?*pattern*

Search backward in the file for the *n*th line containing the pattern. The search starts at the line immediately before the top line displayed.

Certain characters are special at the beginning of the pattern; see the / command.

[*n*]Esc */pattern*

Same as /*.

[*n*]Esc ?*pattern*

Same as ?*.

[*n*] n

Repeat the previous search, for the *n*th line containing the last pattern. If the previous search was modified by !, the search is made for the *n*th line

Copyright © 2014, QNX Software Systems Limited

not containing the pattern. If the previous search was modified by *, the search continues in the next (or previous) file if not satisfied in the current file. There's no effect if the previous search was modified by @.

[*n*] N

Repeat previous search, but in the reverse direction.

Esc n

Repeat previous search, but crossing file boundaries. The effect is as if the previous search were modified by *.

Esc N

Repeat previous search, but in the reverse direction and crossing file boundaries.

:e [*filename*]

Examine a new file. If the filename is missing, the "current" file (see the :n and :p commands) from the list of files in the command line is reexamined. A percent sign (%) in *filename* is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. The filename is inserted into the command-line list of files so that it can be seen by subsequent :n and :p commands. If the filename consists of several files, they're all inserted into the list of files and the first one is examined.

E

Same as :e.

[*n*] :n

Examine the next file (from the list of files given in the command line). If a number *n* is specified, the *n*th next file is examined.

[*n*] :p

Examine the previous file in the command-line list. If a number *n* is specified, the *n*th previous file is examined.

[*n*] :x

Examine the first file in the command-line list. If a number *n* is specified, the *n*th file in the list is examined.

=

Copyright © 2014, QNX Software Systems Limited

Print some information about the file being viewed, including its name and the line number and byte offset of the bottom line being displayed. If possible, this also prints the length of the file, the number of lines in the file, and the percent of the file above the last displayed line.

-char

Followed by one of the command-line option letters, this command changes the setting of that option and prints a message describing the new setting. If the option letter has a numeric value (such as -b or -h), or a string value (such as -P or -t), a new value may be entered after the option letter. If no new value is entered, a message describing the current setting is printed and nothing is changed.

-+char

Followed by one of the command-line option letters, this command resets the option to its default setting and prints a message describing the new setting. The `-+X` command does the same thing as `-+X` on the command line. This doesn't work for string-valued options.

--char

Followed by one of the command-line option letters, this command resets the option to the *opposite* of its default setting and prints a message describing the new setting. The `--X` command does the same thing as -X on the command line. This doesn't work for numeric or string-valued options.

+cmd

Causes the specified *cmd* to be executed each time a new file is examined. For example, `+G` causes `less` to initially display each file starting at the end rather than the beginning.

V

Print the version number of `less` being run.

q

Exit `less`.

v

Invoke an editor to edit the current file being viewed. The editor is taken from the environment variable ***EDITOR***, or defaults to `vedit`.

! ***shell_command***

Copyright © 2014, QNX Software Systems Limited

Invoke a shell to run the *shell_command* given. A percent sign (%) in the command is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file.

`!!` repeats the last shell command, and `!` with no shell command simply invokes a shell. In all cases, the shell is taken from the environment variable ***SHELL***, or defaults to *sh* (p. 1760).

| *m shell_command*

The *m* represents any mark letter. Pipe a section of the input file to the given shell command. The section of the file to be piped is between the current position and the position marked by the letter. The *m* may also be ^ or $ to indicate the beginning or end of the file. If *m* is a dot (`.`) or newline, the current screen is piped. The current screen is the minimum amount piped in any case.

Files:

`/usr/local/lib/less.hlp`

Help file for `less`'s `help` command.

Environment variables:

COLUMNS

Sets the number of columns on the screen. Takes precedence over the number of columns specified by the ***TERM*** variable.

EDITOR

The name of the editor (used for the `v` command).

LESS

Options that are passed to `less` automatically.

LESSEDIT

Editor prototype string (used for the `v` command).

LINES

Sets the number of lines on the screen. Takes precedence over the number of lines specified by the ***TERM*** variable.

SHELL

The shell used to execute the `!` command, as well as to expand filenames.

Copyright © 2014, QNX Software Systems Limited

TERM

The type of terminal on which `less` is being run. ***TERM*** *must* be set.

TMPDIR

Overrides the default location for temporary files (`/tmp`). The `/tmp` directory or the one specified by ***TMPDIR*** *must* be a writable filesystem.

Contributing author:

Mark Nudelman

Caveats:

The = command reports the line number of the line at the top of the screen, but the byte and percent of the line at the bottom of the screen.

If the `:e` command is used to name more than one file and if one of the named files has been viewed previously, the new files may be entered into the list in an unexpected order.

Copyright © 2014, QNX Software Systems Limited

link

Create a link to a file (POSIX)

Syntax:

```
link existing new
```

Runs on:

QNX Neutrino

Options:

existing

Pathname of an existing file.

new

Pathname of the link to be created.

Description:

The `link` utility creates a hard link, *new*, that points to another file, *existing*. To create a symbolic link, use *ln* (p. 1114).

Exit status:

0

Successful completion.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

ln

Create links to (aliases for) files (POSIX)

Syntax:

```
ln [-f|-i] [-Psv] source_file target_file
ln [-f|-i] [-Psv] source_file... target_dir
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-f

Force existing destination pathnames to be removed before linking; don't prompt for confirmation.

-i

(QNX Neutrino extension) Run interactively; write a prompt to the standard error output requesting confirmation for each link that would overwrite an existing file.

-P

Create the link in the process manager's in-memory prefix tree.

-s

Create a symbolic link.

-v

Verbose. Write actions performed to standard output.

source_file

The pathname of a file to be linked. If -s is specified, this file need not exist.

target_file

The pathname of the new directory entry to be created.

target_dir

The pathname of an existing directory in which the new directory entries are to be created.

Copyright © 2014, QNX Software Systems Limited

Description:

The `ln` utility has two syntax forms, as follows:

`ln` [-fl-i] [-s] *source_file target_file*

The `ln` utility creates a new directory entry (link) at the destination path specified by the *target_file* operand which points to the *source_file* as a hard or symbolic link, depending on the -s option. This syntax form is assumed when the destination path *doesn't* name an existing directory.

`ln` [-fl-i] [-s] *source_file... target_dir*

For each *source_file*, `ln` creates a new directory entry at a destination path in the existing directory named by the *target_dir* operand.

The destination path for each *source_file* is the same as its basename (final path component). For example:

```
ln dir/dir/myfile /existingdir
```

creates `/existingdir/myfile` as a link to `dir/dir/myfile`.

This second syntax form is assumed when either the destination names an existing directory, or when more than one source file is specified.

If the destination path exists and you have write permission for the existing destination file, or if -f was specified, `ln` unlinks the destination, then creates the new link.

If you don't have write permission for an existing directory path, and -f isn't specified, and if the standard input is a tty, `ln` prompts you for confirmation prior to unlinking the existing file. If standard input isn't a tty, `ln` writes a diagnostic message to standard error, and goes on to the next *source_file* without unlinking the destination file.

To create the new link, or replace a file with a link, you need write permissions for the directory in which the new link is going to reside. Note that `root` always has this permission regardless of the file permission settings.

Hard links are limited to within the same filesystem as the original file and aren't permitted on directories. With symbolic links, however, you can link *any* pathname to a file. A symbolic link is a special file that has the destination pathname as its data. For more information, see the section on symbolic links in *System Architecture*.

If the -P option is specified, the link is created within the pathname prefix tree maintained in memory by the QNX Neutrino process manager, *procnto* (p. 1586). This lets you create new pathname links without the need for a traditional filesystem. If the -s option is specified, a symbolic redirection takes place. If -s isn't specified, a direct link to a named resource manager is made. The resource manager named in the source must be of the form:

nid,*pid*,*chid*,*handle*

Copyright © 2014, QNX Software Systems Limited

where

nid

Node ID of resource manager to link to.

pid

Process ID of resource manager to link to.

chid

Channel ID of resource manager to link to.

handle

Handle of a pathname prefix of the resource manager to link to.

Most prefix links are symbolic.

You can get some surprising results if you create a symbolic link to a directory that's a symbolic link to somewhere else. For more information, see "Symbolic links" in the Working with Filesystems chapter of the QNX Neutrino *User's Guide*.

Examples:

Create a link to `/home/curious/monkey` called `gorilla` in the `/home/george` directory:

```
ln /home/curious/monkey /home/george/gorilla
```

Create a symbolic link to the directory `/home/fred` called `/home/barney`:

```
ln -s  /home/fred /home/barney
```

Create a symbolic prefix link to `/dev/shmem` from `/tmp`. This simple link maps all temporary files created under the `/tmp` directory into shared memory objects, thus implementing a RAM disk.

```
ln -sP /dev/shmem /tmp
```

Create a symbolic prefix to `/dev/ser1` from `/dev/modem`. Any attempt to open `/dev/modem` results in an open of `/dev/ser1`.

```
ln -sP /dev/ser1 /dev/modem
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

All the specified files were linked successfully.

>0

An error occurred.

Caveats:

When creating a symbolic link, `ln` doesn't check that the *source_file* named actually exists, or even that it's a valid pathname. If the file doesn't exist or if the *source_file* isn't a valid pathname, any attempts to use the link fail.

If a destination path exists, and `ln` is interrupted before ending, the destination path may be removed before the new link is created.

Copyright © 2014, QNX Software Systems Limited

ln-w

Create a "link" on Windows

Syntax:

```
ln-w [-s] source [destination]
```

Runs on:

Microsoft Windows

Options:

-s

Create a symbolic link.

Description:

The `ln-w` utility pretends to create a link on Windows machines; it actually copies the given file to the specified destination. If you don't specify a *destination*, `ln-w` copies the specified file to the *basename* (p. 54) of the *source*.

Copyright © 2014, QNX Software Systems Limited

logger

Make entries in the system log (POSIX)

If you aren't `root`, specify the full path: `/usr/sbin/logger`.

Syntax:

```
logger [-is] [-f file] [-p pri] [-t tag]
       [message ...]
```

Runs on:

QNX Neutrino

Options:

-f *file*

Log the specified *file*.

-i

Log the process ID of the logger process with each line.

-p *pri*

Enter the message with the specified *priority*. The priority may be specified numerically or as a *facility*`.`*level* pair. For example, -p local3.info logs the message(s) as `informational` level in the `local3` facility. The default is user.notice.

-s

Log the message to standard error, as well as the system log.

-t *tag*

Mark every line in the log with the specified *tag*.

message

Write the *message* to log; if not specified, and the -f flag isn't provided, standard input is logged.

Copyright © 2014, QNX Software Systems Limited

Description:

The `logger` command provides a shell command interface to the *`syslogd`* (p. 1885) daemon.

Examples:

Log the message “System rebooted”:

```
logger System rebooted
```

Log the file `/tmp/log` tagging each line with `log`:

```
logger -f /tmp/log -t log
```

Files:

`/usr/sbin/logger`

The `logger` utility is located in the `/usr/sbin/` directory, which is not included in the default ***PATH*** of non-root users. If you are not `root`, specify the full path.

Environment variables:

SYSLOG

When defined, ***SYSLOG*** specifies which node `syslogd` is running on. By default, the local node is assumed.

Exit status:

0

Successful completion.

>0

An error occurred.

Because the syslog API doesn't return error codes, only argument errors can be detected.

Copyright © 2014, QNX Software Systems Limited

login

Log in

Syntax:

```
login [-fpq] [-c|u] [-h host] [-t timeout]
      [username [env1 env2 ...]]
```

Runs on:

QNX Neutrino

Options:

-c

Use only a QNX 4-compatible encryption.

-f

Skip login authentication (force). This option is used to indicate that proper authentication has already been done and no password is required. If you're `root`, it allows you to log in as *username* without a password prompt. You can use this option only if you're logged in as `root` or you're already logged in as *username*.

-h *host*

Specify for login purposes that you're connecting from *host*.

-p

Preserve the environment. If -p isn't specified, the environment is cleared, and only the following variables are set: ***SHELL***, ***HOME***, ***LOGNAME***, ***USERNAME*** (same as ***LOGNAME***), ***TERM*** (if set in the caller's environment), and ***PATH*** (set to the value defined by _CS_PATH).

The environment variables set in `/etc/default/login` are set whether or not -p is specified.

-q

Be quiet; disable extra output messages.

-t *timeout*

Cancel the login if no response is received within *timeout* number of seconds. A value of 0 (the default) disables the timeout facility.

Copyright © 2014, QNX Software Systems Limited

-u

Use only a UNIX-compatible encryption.

Description:

The `login` command starts a session associated with a user. It's used when someone either first signs on to the computer, or wishes to log in as another user. If `login` is invoked without a *username*, it prompts for one. It always prompts for a password.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

The `login` command verifies the username and password against a password database and, if verified, starts the user session. If either the username or password is unacceptable, `login` asks again for both.

For compatibility, `login` checks the password using two encryption methods:

qnx_crypt()

Compatible with QNX 4.

crypt()

Compatible with UNIX (NetBSD, Linux, Solaris).

The check succeeds if either encryption matches the password. The -c and -u options allow slightly more security by allowing only one encryption method.

The `login` utility sets the user ID and group ID as well as the current working directory, then executes a shell according to specifications found in the password database.

If execution of the shell fails, `login` prints the message `No Shell` and exits. The shell is started with a dash (-) prepended to its name as argument 0. This informs the shell to perform its startup routine.

The `login` utility sets the ***SHELL***, ***HOME***, ***LOGNAME***, ***USERNAME*** (same as ***LOGNAME***), ***TERM*** (if set in the caller's environment), and ***PATH*** (set to the value defined by _CS_PATH) environment variables. For more information on setting ***PATH***, see "Setting ***PATH*** and ***LD_LIBRARY_PATH***" in the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*.

The `login` utility also updates system accounting information in `/var/log/utmp`, `/var/log/wtmp`, and `/var/log/lastlog` if they already exist.

Copyright © 2014, QNX Software Systems Limited

The `login` utility doesn't create `/var/log/utmp`, `/var/log/wtmp`, and `/var/log/lastlog` if they don't already exist. These files can quickly become very big, which isn't good on an embedded system with limited resources.

The `login` utility also sets the environment variables specified in the `/etc/default/login` file. This file lets you specify which environment variable settings are to be used across login sessions. If a variable isn't set when `login` is started and this file contains a default for the variable, that default is used. If the file doesn't contain a default for the variable, the variable is cleared by `login`. If a variable is already set when `login` is started and -p is specified, it's preserved, even if the `/etc/default/login` file contains a default for the variable. The -p option allows site-specific environment variables to be passed from the system startup processes down to applications.

You can set environment variables by specifying them on the command line after the username, in the form *VARIABLE*[=*VALUE*]. For example:

```
login thomasf APPLES=RED TOMATOES
```

adds the following to the new login environment:

```
APPLES = RED
TOMATOES = 1
```

Values passed on the command line are always added to the new environment (regardless of the -p flag) and override the existing values. The ***SHELL***, ***HOME***, ***TERM***, ***LOGNAME***, ***USER*** values set by `login` override the command-line values.

You can specify which environment variables are to be preserved when the -p flag isn't specified, by populating an `/etc/default/login` file with the names of the environment variables you want to save. For example, if `/etc/default/login` contains:

```
SYSNAME
TZ
```

then when you run `login` without the -p option, the ***SYSNAME*** and ***TZ*** variables (if they exist) are propagated to the new environment. The values may be overwritten by values on the command line.

You can execute `login` only from the login shell, or when no session is active; you can't nest `login` commands.

If you execute `login` from the login shell, the shell from which it is run is terminated. This is because, by default, `login` is an alias for `exec login`.

Files:

Copyright © 2014, QNX Software Systems Limited

/bin/sh

The Korn shell command interpreter.

/dev/console

Refers to the system console device.

/dev/log

A FIFO that receives log messages for `syslogd`.

/dev/tty

Refers to the controlling terminal for the current process.

/etc/group

Defines the known groups for the system.

/etc/default/login

Sets login environment variables.

/etc/passwd

Contains the user database (username, user ID, group ID, full name, home directory, shell).

/etc/shadow

Contains encrypted login password for each userid.

/var/log/utmp, /var/log/wtmp, /var/log/lastlog

Accounting records are appended to these files if they already exist.

Copyright © 2014, QNX Software Systems Limited

logout

Terminate a session (UNIX)

Syntax:

```
logout
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `logout` command terminates a session. You can execute this command only from a shell invoked by a *login* (p. 1121) command; otherwise a message is printed and the `logout` command is ignored.

Copyright © 2014, QNX Software Systems Limited

lpd

Line printer spooler daemon

Syntax:

```
lpd [-ln] [portnum]
```

Runs on:

QNX Neutrino

Options:

-l

("el") Log valid requests received from the network. This can be useful for debugging purposes.

-n

Don't check to see if the job host is included in `/etc/hosts.equiv` or `/etc/hosts.lpd`. This option allows anyone on a network to print.

portnum

Although the Internet port number used to rendezvous with other processes is normally obtained with *getservbyname()*, you can use this option to change the port number.

Description:

The `lpd` daemon makes a single pass through the `printcap` database, restarting any printers that have jobs. The daemon listens for requests to:

- print files in the queue
- transfer files to the spooling area
- display the queue
- remove jobs from the queue.

In each case, `lpd` forks a child to handle the request, so that the parent can continue to listen for more requests.

Access control is provided by two means:

- All requests must come from one of the machines listed in the file `/etc/hosts.equiv` or `/etc/hosts.lpd`.

Copyright © 2014, QNX Software Systems Limited

- If the `rs` capability is specified in the `printcap` entry for the printer being accessed, `lpr` requests are honored only for those users with accounts on the machine with the printer.

The `lpd` daemon uses simple text files as lock files for synchronization. The parent daemon uses the file `/usr/spool/output/lpd.lock`, while its children use a `.lock` file within each spool directory specified in the `printcap` file.

Both the `/usr/spool/output` and `/etc/printcap` directories must exist, or `lpd` won't run. If these directories exist, and `lpd` still won't run, remove `/usr/spool/output/lpd.lock` if it exists (e.g after a power failure or system crash).

The lock file is kept in a readable ASCII form and contains two lines. The first line is the *pid* of the daemon who owns the lock. The second line of the child's lock file contains the current job or status.

To keep a printer from filling your hard disk, a `minfree` file may be created in its spool directory. This file should contain the number of blocks (in ASCII) to leave free.

If errors occur, `lpd` writes messages to the system log. In order to capture the log messages, you need to have *`syslogd`* (p. 1885) running.

If errors occur, `lpd` doesn't send much information to standard error, but it saves lots of information in the system log. If you have problems with `lpd`, the system log will very likely help you figure out what's wrong.

Files:

`/etc/printcap` **(p. 1575)**

Printer description file.

`/usr/spool/*`

Spool directories.

`/usr/spool/*/minfree`

Minimum free space to leave.

`/etc/hosts.equiv` **(p. 947)**

A list of machine names that are allowed to access the printers.

`/etc/hosts.lpd`

A list of machine names that are allowed to access the printers, but not under the same administrative control.

Copyright © 2014, QNX Software Systems Limited

lpr

Print on a line printer

Syntax:

```
lpr [-#num] [-1234font] [-cdfghlmnprstv] [-C class]
    [-i [numcols]] [-J job] [-Pprinter] [-T title]
    [-U user] [-wnum] [filename...]
```

Runs on:

QNX Neutrino

Options:

-#num

The number of copies desired of each file named. For example:

```
lpr -#3 foo.c bar.c more.c
```

results in three copies of the file `foo.c`, followed by three copies of the file `bar.c`, etc. On the other hand:

```
cat foo.c bar.c more.c | lpr -#3
```

gives three copies of the concatenation of the files. Often a site disables this feature to encourage use of a photocopier instead.

-[1234]*font*

Specifies a font to be mounted on font position *i*.

-C *class*

Job classification to use on the burst page. For example:

```
lpr -C EECS foo.c
```

causes the system name (the name returned by *hostname*) to be replaced on the burst page by EECS, and the file `foo.c` to be printed.

-c

The files are assumed to contain data produced by `cifplot`.

Copyright © 2014, QNX Software Systems Limited

-d

The files are assumed to contain data from tex.

-f

Use a filter that interprets the first character of each line as a standard FORTRAN carriage control character.

-g

The files are assumed to contain standard plot data.

-h

Suppress the printing of the burst page.

-i [*numcols*]

Indent the output. If the next argument is numeric (*numcols*), it's used as the number of blanks to be printed before each line; otherwise, 8 characters are printed.

-J *job*

The job name to print on the burst page. Normally, the first file's name is used.

-l

Use a filter that lets control characters be printed and suppresses page breaks.

-m

Send mail upon completion.

-n

The files are assumed to contain data from `ditroff`.

-P*printer*

Force output to a specific printer. The *printer* argument must be a printer name that's defined in */etc/printcap* (p. 1575). Normally, `lpr` uses the (site-dependent) default printer, or the one specified by the ***PRINTER*** environment variable.

-p

Use *pr* (p. 1571) to format the files.

-r

Copyright © 2014, QNX Software Systems Limited

Remove the file upon completion.

-s

Use symbolic links.

-T *title*

The title name for `pr` (instead of the filename).

-t

The files are assumed to contain data from `troff`.

-U *user*

The user name to print on the burst page, also for accounting purposes. This option is honored only if the real userid is `daemon` (or that specified in the `/etc/printcap` file instead of `daemon`), and is intended for those instances where print filters wish to re-queue jobs.

-v

The files are assumed to contain a raster image.

-w*num*

Use *num* as the page width for `pr`.

filename

The name of the file to print.

Description:

The `lpr` utility uses the spooling daemon `lpd` to print the named files when facilities become available. If no names appear, the standard input is assumed.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *`mkefs`* (p. 1209), *`mketfs`* (p. 1219), or *`mkifs`* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Diagnostics

If a user other than `root` prints a file and spooling is disabled, `lpr` prints a message saying so and won't put jobs in the queue.

If a connection to `lpd` on the local machine can't be made, `lpr` says that the daemon can't be started.

Copyright © 2014, QNX Software Systems Limited

Environment variables:

PRINTER

An alternate default printer.

DONT_USE_LINK_UNLINK

Use *rename()* instead of *link()* or *unlink()*.

Caveats:

If you try to spool a file that's too large, it's truncated.

If the `lpd` daemon has any problems (e.g. it can't find the spool file), it might print some diagnostics in its log file.

Copyright © 2014, QNX Software Systems Limited

lprc

Control the line printers

Syntax:

```
lprc [command [argument ...]]
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The system administrator (i.e. `root`) uses the `lprc` utility to control the operation of the line printers that are configured in `/etc/printcap`. You can use `lprc` to:

- disable or enable a printer
- disable or enable a printer's spooling queue
- rearrange the order of jobs in a spooling queue
- find the status of printers, and their associated spooling queues and printer daemons.

Without any arguments, `lprc` prompts for commands from the standard input. If arguments are supplied, `lprc` interprets the first argument as a command and the remaining arguments as parameters to the command. You can redirect standard input so that `lprc` reads commands from file. You can abbreviate the commands.

Here's a list of recognized commands:

? [*command ...*] or help [*command ...*]

Print a short description of each command specified in the argument list. If you don't specify a command, `lprc` displays a list of the recognized commands.

abort { all | printer }

Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by `lpr`) for the specified printers.

clean { all | printer }

Copyright © 2014, QNX Software Systems Limited

Remove any temporary files, data files, and control files that can't be printed (i.e. that don't form a complete printer job) from the specified printer queue(s) on the local machine.

disable { all | printer }

Turn the specified printer queues off. This prevents `lpr` from adding new to the queue.

down { all | printer } ***message ...***

Turn the specified printer queue off, disable printing, and put *message* in the printer status file. The message doesn't need to be quoted; the remaining arguments are treated like `echo`.

You normally use the `down` to take a printer down, let others know why `lprq` indicates the printer is down, and print the status message.

enable { all | printer }

Enable spooling on the local queue for the listed printers. This lets `lpr` put new jobs in the spool queue.

exit or **quit**

Exit from `lprc`.

restart { all | printer }

Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly, leaving jobs in the queue. The `lprq` utility reports that no daemon is present when this condition occurs. If the user is the superuser, try to abort the current daemon first (i.e. kill and restart a "stuck" daemon).

start { all | printer }

Enable printing and start a spooling daemon for the listed printers.

status [all | printer]

Display the status of daemons and queues. If you don't specify a printer, `lprc` displays the status of all printers defined in the `/etc/printcap` file.

stop { all | printer }

Stop a spooling daemon after the current job completes and disable printing.

topq printer [*jobnum* ...] [*user* ...]

Place the jobs in the order listed at the top of the printer queue.

Copyright © 2014, QNX Software Systems Limited

up { all | printer }

Enable everything and start a new printer daemon. This command undoes the effects of down.

Files:

/etc/printcap **(p. 1575)**

Printer description file.

Errors:

?Ambiguous command

Abbreviation matches more than one command.

?Invalid command

No match was found.

?Privileged command

The command can be executed only by root.

Copyright © 2014, QNX Software Systems Limited

lprq

Examine a line printer's spool queue

Syntax:

```
lprq [-l] [-Pprinter] [jobnum ...] [user ...]
```

Runs on:

QNX Neutrino

Options:

-l

Print information about each of the files comprising the job entry. Normally, `lprq` displays only as much information as fits on one line.

-Pprinter

The printer to query. If you don't specify a printer, `lprq` queries the default line printer (or the printer identified by the ***PRINTER*** environment variable, if it's set). All other arguments supplied are interpreted as user names or job numbers to filter out only those jobs of interest.

jobnum

For each job submitted (i.e. invocation of *lpr* (p. 1128)), `lprq` reports the user's name, current rank in the queue, the names of files comprising the job, the job identifier (a number that you can give to *lprrm* (p. 1137) to remove a specific job), and the total size in bytes.

The order of the jobs depends on the algorithm used to scan the spooling directory and is supposed to be FIFO (First in First Out).

Filenames comprising a job may be unavailable (when `lpr` is used as a sink in a pipeline), in which case the file is indicated as “standard input.”

If `lprq` warns that there's no daemon present (i.e. due to some malfunction), you can use *lprc* (p. 1132) to restart the printer daemon.

user

Specify a particular user name.

Copyright © 2014, QNX Software Systems Limited

Description:

The `lprq` utility examines the spooling area used by `lpd` for printing files on the line printer, and reports the status of the specified jobs or all jobs associated with a user. If you invoke `lprq` without any arguments, it lists any jobs that are currently in the queue.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Files:

/etc/printcap **(p. 1575)**

Printer description file.

Environment variables:

PRINTER

An alternate default printer.

Caveats:

Due to the dynamic nature of the information in the spooling directory, `lprq` may report unreliably. Output formatting is sensitive to the line length of the terminal; this can result in widely spaced columns.

Copyright © 2014, QNX Software Systems Limited

lprrm

Remove jobs from a line printer's spooling queue

Syntax:

```
lprrm [-Pprinter] [-] [jobnum ...] [user ...]
```

Runs on:

QNX Neutrino

Options:

-Pprinter

Specify the queue associated with a specific *printer*; otherwise, the default printer is used.

-

Remove *all* of the jobs that you own. If the superuser (`root`) uses this flag, `lprrm` empties the entire spool queue.

jobnum

Remove the individual job specified by the given job number. You can get this number from the *lprq* (p. 1135) utility.

user

Attempt to remove any jobs queued belonging to that user (or users). This form of invoking `lprrm` is useful only to the superuser.

Description:

The `lprrm` utility removes one or more jobs from a printer's spool queue. Since the spooling directory itself is protected from users, `lprrm` is normally the only way that you can remove a job. The job's owner is determined by the user's login name and hostname on the machine where the `lpr` command was invoked.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Copyright © 2014, QNX Software Systems Limited

Note that if you don't specify any arguments or options, `lprrm` deletes the currently active job if you own it.

The `lprrm` utility announces the names of any files it removes and is silent if the queue contains no jobs that match the request list.

The utility kills off an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removals.

Files:

/etc/printcap **(p. 1575)**

Printer description file.

Environment variables:

PRINTER

An alternate default printer.

Errors:

`Permission denied`

Displayed when you try to remove print jobs that aren't yours.

Caveats:

Since there are race conditions possible in the update of the lock file, the currently active job may be incorrectly identified.

Copyright © 2014, QNX Software Systems Limited

ls

List directory contents (POSIX)

Syntax:

```
ls [-1CFRacdilqrstu] [-DLSbfghnopv] [file...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-1

("One") Force output to be one entry per line.

-a

List all files, including hidden ones i.e., those that start with a dot (.). By default, these entries aren't listed.

-C

Display multiple-column output, with entries sorted down the columns according to the collating sequence.

-c

For sorting (-t) or printing (-l), use time of last change to the file's status information instead of time of last modification of the file itself.

-d

Treat directories like files — give information on the directory itself, not on the files or subdirectories it contains.

-F

Indicate the filetype by adding an extra character after some pathnames, as follows:

Character	Meaning
/	Directories
*	Executable files
|	FIFOs (named pipes)

Copyright © 2014, QNX Software Systems Limited

Character	Meaning
#	Named special files (QNX Neutrino extension)
@	Symbolic links (QNX Neutrino extension)

-i

For each file, print the file's serial number.

-l

("el") List in long format. This option provides most of the relevant file information (filetype, permissions, link count, owner/group of the file, as well as the size, date, and name of the file), as follows:

```
drwxrwxrwx  7    0        0       22528   Jan 17 15:38 Csrc
-rw-rw-rw-  1    0        0          22   Feb 14 13:41 barney
-rwxrwxrwx  1    0        0          22   Feb 14 13:41 exec
-rw-rw-rw-  1    0        0          22   Feb 14 13:41 fred
drwxrwxrwx  2    0        0       23040   Feb 12 10:56 libtests
drwxrwxrwx  2    0        0        2048   Sep 28 06:39 util
```

To see a header above the columns, use the -h option.

Note that the initial field (e.g., "`drwxrwxrwx`") describes the filetype and permissions (see *below* (p. 1143)).

-q

Force filename characters that aren't included in the character set classification in the current locale to be displayed as a question mark (?). This is the default if output is to a terminal.

QNX Neutrino currently supports only the POSIX (i.e., C) locale.

-R

Recursively list all subdirectories encountered.

If you're using Qnet, doing something like `ls -R /net` can take a very long time because it recursively lists all the directories on all the machines on your network.

-r

Reverse the order of the sort to get either oldest files first (if files are being sorted by time) or reverse collating sequence.

Copyright © 2014, QNX Software Systems Limited

-s

Display the size of the file in 512-byte blocks.

-t

Sort by time modified (most recently modified first) before sorting the files by the collating sequence.

-u

For sorting (-t) or printing (-l), use time of last access (i.e., last use), instead of time of last modification of the file.

file

The pathname of a file to be listed. If the file specified isn't found, a diagnostic message is output on standard error. Files are displayed in command-line sequence.

QNX Neutrino extensions:

-b

Use the size of the file for sorting and printing. Sort in descending order.

-D

Display directories only.

-f

Don't sort the output (same as -S).

-g

List in long format, as in `ls -l`, but don't show owner (group is displayed).

-h

Display a header for the -l ("el") and -n options.

-L

Resolve symbolic links instead of showing them.

-n

Same as -l ("el"), except display group `ID` and user `ID` numbers instead of names.

-o

List in long format, as in `ls -l`, but don't show group (owner is displayed).

Copyright © 2014, QNX Software Systems Limited

-p

Display a list of relative pathnames for all nondirectory files. The files are listed one to a line. This option allows you to pass full pathnames of files to programs.

-S

Don't sort the output. This option is useful for determining the order in which entries are found in a given directory.

-v

List directories first.

Description:

For each file you name that isn't a directory, `ls` displays the file's name as well as any information requested on the file.

For each directory you name, `ls` displays the names of files contained within that directory, as well as any information requested on the files. The -d option overrides this behavior and makes `ls` display information on the directory itself, rather than on its contents.

If you specify more than one file, `ls` displays files that aren't directories first. Directories and nondirectories are sorted separately.

If you don't specify a file, `ls` displays the contents of the current directory.

Specifying more than one of the -C, -l ("el"), and -1 ("one") options isn't considered an error. The last option specified determines the output format.

In many environments, the `ls` command is "aliased" to either `ls -C` or `ls -CF`, the two most common `ls` display formats. Unless the ***POSIX_STRICT*** environment variable is set, `ls` defaults to the multi-column output (-C option).

The -p option is useful for passing a list of all nondirectory filenames, one filename per line, to other programs. The filenames include the full pathnames.

When displaying a timestamp for a file, `ls` displays the date and time, unless the file is older or newer than the current date by six months (a "month" is defined as 30 days). Otherwise, `ls` displays the date and year.

If you use the -l ("el") or -s option — or the -n, -g, or -o options, which imply them — and you list a directory, the output includes the total number of 512-byte blocks that the directory occupies. This total doesn't include the space occupied by any subdirectories. For example:

```
$ ls -l /etc/rc.d
total 28
-rwxrwxr-x  1 root      root           1515 Apr 30  2001 rc.devices
-rwxrwxr-x  1 root      root            354 Apr 08 14:37 rc.local
```


Copyright © 2014, QNX Software Systems Limited

```
-rwxrwxr-x  1 root      root           321 Dec 23  2004 rc.local~
-rwxrwxr-x  1 root      root          6767 May 31  2001 rc.setup-info
-rwxrwxr-x  1 root      root          2993 May 07  2002 rc.setup-once
-rwxrwxr-x  1 root      root          1271 Apr 21  2002 rc.sysinit
```

Filetype and permissions

In the long format (-l option), the initial field (e.g., "drwxrwxrwx") describes the filetype and permissions. The first character represents the filetype; the next nine represent the read/write/executable permissions for the owner, group, and other classes.

If there's a plus sign (+) after the permissions, the file also has an access control list. For more information, see *getfacl* (p. 903) and *setfacl* (p. 1747).

In the first position, the following characters are used to indicate the filetype:

Character	Meaning
-	Regular file
b	Block special file
c	Character special file
d	Directory
l	Symbolic link
n	Named special file
p	FIFO (pipe)
s	Unix domain socket

The next nine characters represent the owner, group, and other permissions; each class has a three-character field. For each class, the characters and the first two positions are as follows:

Position	Character	Meaning
First	r or -	The file is readable or not
Second	w or -	The file is writable or not

For the third position, several characters are possible:

Character	Meaning
S	If a file, it isn't executable and is *setuid* (if in the owner field) or *setgid* (if in the group field). If a directory, it isn't searchable. All files within that directory inherit the

Copyright © 2014, QNX Software Systems Limited

Character	Meaning
	permissions of the directory, *not* of the creator of the files.
s	The file is executable or the directory is searchable. The user/group ID modes are set, and the directory-driven inheritance is as with S.
T	Sticky bit is set and x isn't set.
t	Sticky bit is set and x is set.
x	The file is executable or the directory is searchable.
-	None of the attributes (S, s, T, t, or x) applies.

Environment variables:

COLUMNS

If this variable contains a string representing a decimal integer, it indicates the user's preferred column position width for displaying multiple-column output. The `ls` utility calculates how many pathname text columns to display (see -C) based on the width provided. If ***COLUMNS*** isn't set or is invalid, the number of columns displayed is determined by the type of output device.

POSIX_STRICT

Interpret options according to POSIX specifications.

TZ

Determines the time zone for date and time displays.

Exit status:

0

All files were listed successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

lsm-autoip.so

AutoIP negotiation module for link-local addresses

Syntax:

```
mount -Tio-pkt [-o option,option,...] lsm-autoip.so
```

or:

```
io-pkt-variant -p autoip ... &
```

where *variant* is one of `v4`, `v4-hc`, or `v6-hc`. If you use *mount* (p. 1312), you can specify a special device of `tcpip` or `io-pkt`.

Runs on:

QNX Neutrino

Options:

if=*interface*

The interface on which to start the AutoIP service. Only one interface is supported at one time. The default is `en0`.

ip=*address*

Attempt to claim *address* (in dotted numeric notation) for the link-local address. If this fails, a random address is attempted until an unused IP address is found. You can use this option to force the AutoIP module to use a certain initial IP address. The default random IP address is based on the MAC address of the interface. The same initial IP address would be chosen for a specific MAC address.

debug

Enable debugging in verbose mode. Debug messages are logged to *slogger* (p. 1807).

abandon

Abandon the chosen link-local address if challenged. By default, the AutoIP module attempts to defend the link-local address that it chose, challenging other hosts that use the same address. If the other host persists, the AutoIP module releases the IP address, and chooses another. If you use this option, the AutoIP module releases the IP address as soon as a conflict is detected.

Copyright © 2014, QNX Software Systems Limited

delay=*ms*

Set the delay before packets are transmitted. This affects PROBE packets only. The delay is specified in milliseconds. You can set this option for a range of 8000 to 1 ms. The default is 2000 ms. You may want to set a shorter delay if your network isn't connected with Ethernet switches.

force

Even if the interface obtained is a routable IP, force a link-local IP address. The default behavior is to leave the link-local IP untouched; see the Caveats below.

The force option is ignored if you haven't specified the old option as well.

old

Attempt to manage the link-local network using the routing table. For example, route the link-local subnet via the routable IP address if it exists. The default is to have the link-local and routable IP address behave as aliases and co-exist.

Description:

The AutoIP module (`lsm-autoip.so`) configures a specified interface with a link-local IP address by negotiating with neighboring hosts. If no host on the local network is using the IP address that the module has chosen, the interface is configured with the chosen address.

The AutoIP module chooses its address from the IANA registered IP address network of 169.254/16. Some of this network is reserved for special purposes, so the available addresses are from 169.254.1.0 to 169.254.254.255.

Once an IP address is chosen and configured, the AutoIP module continues to monitor the network for address conflicts, and either defends or changes the address assigned to the interface to correct conflicts.

Only one interface can be supported at one time. The module can be loaded only once.

The interface that AutoIP is servicing must exist before the AutoIP module is loaded.

When the AutoIP module configures a TCP/IP interface with an IP address, it does so as an alias. This way, if the interface already has an IP address, it won't delete the primary or first IP address; the interface is assigned with both addresses (and potentially more if needed). This allows the module to coexist with other modules or applications that configure TCP/IP stack interfaces, such as *`dhcp.client`* (p. 544) (when used with the -a option).

Copyright © 2014, QNX Software Systems Limited

When more than one IP address is assigned to an interface, it's considered an alias, and must be referenced directly if it needs to be deleted; otherwise you may delete the first IP address in the list of addresses, instead of the address you intended to delete. For example, if you're manually configuring an interface in combination with AutoIP, type something like this:

```
ifconfig en0 alias 10.0.0.1
```

This way, you won't replace an IP address that may already be configured for the interface.

To unconfigure the interface, type:

```
ifconfig en0 delete 10.0.0.1
```

This way, you specifically delete the configuration you previously set. If you execute:

```
ifconfig en0 delete
```

you'll delete the first IP address in the list (whatever this is), instead of the IP address you previously configured above, which may be what the AutoIP module configured.

Examples:

The following command line mounts the AutoIP module to service interface `en0`. The initial link-local IP address that it PROBEs for is 169.254.20.20; it has a 200 ms delay between PROBE packets. If an IP address conflict is detected after the interface has been configured, it releases the address immediately.

```
mount -T io-pkt -oif=en0,ip=169.254.20.20,debug,delay=200,abandon lsm-autoip.so
```

Caveats:

If you're using `dhcp.client` in combination with the AutoIP module, you must start `dhcp.client` with the -a option. This way, `dhcp.client` applies its IP configuration as an alias rather than overwrite the primary configuration.

If you specify the old option, the old AutoIP behavior is enabled, and the AutoIP module will delete the link-local IP address of the interface if it detects that a routable IP address had been configured on the interface. This means that if you already have a connection to other hosts using your link-local IP address, those links will be cut off. If this behavior is a concern, you can use the force option in combination with the old option to force the link-local address to always exist.

Copyright © 2014, QNX Software Systems Limited

lsm-pf-v4.so, lsm-pf-v6.so

Provide IP filter services

Use the version of this module that corresponds to the version of *io-pkt-** (p. 1007) that you're running:

- `lsm-pf-v4.so` with `io-pkt-v4` or `io-pkt-v4-hc`
- `lsm-pf-v6.so` with `io-pkt-v6-hc`

Syntax:

If you're using `io-pkt-v4` or `io-pkt-v4-hc`, use one of the following:

```
io-pkt-v4 -p pf-v4
io-pkt-v4-hc -p pf-v4
mount -Ttcpip lsm-pf-v4.so
```

If you're using `io-pkt-v6-hc`, use one of the following:

```
io-pkt-v6-hc -p pf-v6
mount -Ttcpip lsm-pf-v6.so
```

If you use *mount* (p. 1312), you can specify a special device of `tcpip` or `io-pkt`.

Runs on:

QNX Neutrino

Options:

None.

Description:

The `lsm-pf-v4.so` and `lsm-pf-v6.so` shared objects are the modules that handle IP filtering and NAT (Network Address Translation) services. You need to load these libraries to enable filtering and NAT functionality.

IP filtering allows your host to act as a firewall, or you can provide firewall services on your host. NAT allows multiple hosts on a subnet to share a common IP address.

You use configuration files to set the filtering and NAT rules. For more details, see the documentation for *pf.conf* (p. 1476).

Copyright © 2014, QNX Software Systems Limited

lsm-qnet.so

Transparent Distributed Processing (native QNX Neutrino network) module

Syntax:

```
io-pkt ... -p qnet [option[,option]...]
```

Runs on:

QNX Neutrino

Options:

- The `lsm-qnet.so` DLL has many options to alter how Qnet functions (e.g. timeouts, retries, and idle times), but Qnet is optimized to function with its default settings. Don't use these options unless you're trying to overcome an issue related to your environment. Using these options can have a negative effect if used incorrectly.
- Use commas, not spaces, to separate the options.

auto_add=*X*

When `qnet` starts up, the builtin `io-pkt` Ethernet (*en*) resolver broadcasts its own mappings on all interfaces. This option specifies the time interval, in ticks (see the periodic_ticks option), after which the builtin `io-pkt` Ethernet (en) resolver rebroadcasts its own mappings on all of its interfaces. This has the effect of automatically populating the `/net` directory with all of the connected nodes.

The default is 150 ticks, which represents 30 seconds. The value 0 is special because it not only stops the broadcasted transmissions, but it causes unsolicited received broadcasts to be discarded. This has the effect of populating the `/net` directory only with nodes that applications on the local node specifically open.

bind=en*X*

Specify the interface (e.g. bind=en0) to use. All `/dev/io-net/en`*X* interfaces are used by default. When you specify more than one bind option, Qnet uses all the specified interfaces. This is the fastest packet transport.

Copyright © 2014, QNX Software Systems Limited

The combination of bind=en and resolve=dns is invalid.

bind=*ip*

Instead of using raw (DIX blue-book) ethernet packets, encapsulate Qnet packets with an IP header using its registered protocol number. This is useful on larger networks where simple L2 switching isn't possible, and all packets must be routed.

If you use the bind=`ip` option, you also need to use the resolve=`dns` option. The resolver is used to map the node name to the IP address; you can't use the default resolver with the bind=`ip` option.

conn_est_retries=*X*

The number of times QoS should retry to establish a connection before giving up. The default is 1.

conn_est_timeout=*X*

The number of periodic ticks before QoS should retransmit a connection-establishment request. The default is 1.

conn_up_idle=*X*

The number of periodic ticks until QoS should conclude that a connection is idle without any traffic and should be polled with a heartbeat. The default is 50 ticks (10 seconds).

conn_up_retries=*X*

The number of unanswered connection heartbeats before QoS concludes that a connection is down. The default is 6.

do_crc=*X*

Enable (1) or disable (0) software-level CRC checking of packets by L4. The default is 0. When you disable CRC checking, it yields the best performance on reliable hardware.

enforce_crc=*num*

If you use this option in combination with do_crc, only packets that contain a valid CRC are accepted. This option has an effect only when do_crc is also set to 1. Setting enforce_crc to one causes packets that are received without a valid software-level CRC generated by the remote mode (i.e. it's running do_crc=0) to be discarded, because the packet content's integrity is unknown,

Copyright © 2014, QNX Software Systems Limited

and could be suspect. The default is zero, which allows received packets without a generated software-level CRC to be processed.

host=*hostname*

Change the hostname of the machine.

mapany=*map_uid*

Map any incoming user ID to *map_uid* and map its group ID to that of *map_uid*.

maproot=*map_uid*

If the incoming user ID is 0, map it to *map_uid* and map its group ID to that of *map_uid*.

max_num_l4s=*num*

Specify the number of interfaces. The default is two, and the maximum is four.

max_tx_bufs=*num*

The number of Tx buffers that Qnet holds in reserve before allocating more; the default is 500. If your application sends large messages, you may want to increase this value for performance. If your application typically sends small messages (most default system traffic is small messages), you may want to decrease this value to save memory.

mount=*directory*[:[.]*domain*]]*

Specify a network directory. The default directory is `/net`. The default *domain* is either the hostname domain, if it has one, or the directory with the slashes changed to dots and reversed. For example, `/net/outside/canada` has a domain of `canada.outside.net.` The first mount is the default directory and domain that the local hostname resolves through.

mtu_en=*num*

Specify the maximum transmission unit (MTU) of a Qnet packet. The *num* argument must be greater than 100, and all nodes in network must use the same value. The default is 1500.

no_ack=*X*

Copyright © 2014, QNX Software Systems Limited

Whether or not to generate and expect ACK packets. These packets are required to guarantee data delivery over networks that may lose packets, e.g. Ethernet. The value of *X* can be:

0

Generate and expect ACK packets (the default).

1

Don't generate or expect ACK packets. Specify this value only when it isn't possible for a packet to be lost.

Configure all hosts on the network to use the same value for the no_ack option.

no_slog=*X*

Specifying a nonzero value to this option instructs Qnet not to log errors or events to *slogger* (p. 1807). You can use this option to squelch events caused by a noisy network when you're looking for non-network events in the *sloginfo* (p. 1818) output. By default, Qnet logs events to `slogger`, which corresponds to a zero value for no_slog.

periodic_ticks=*X*

The number of times per seconds that QoS/L4 should wake up to perform periodic housekeeping tasks. The value must be in the range from 1 to 1000; the default is 5, resulting in a default 200 ms tick.

If you change the value of periodic_ticks, you'll affect the timing of all other options that rely on the timer tick.

probe_no_l4_time=*X*

The number of periodic ticks after which QoS should probe a connection on an interface for which there is no mapping for the remote node with a broadcasted packet. The default is the same as the value of conn_up_idle.

qos_per_pri=*num* or qos_tx_pri=*num*

The priority to use for the pulse for the QoS periodic transmission thread and the QoS transmission thread.

Copyright © 2014, QNX Software Systems Limited

We recommend that you not change these values; we supply the qos_per_pri and qos_tx_pri options for the rare cases where you need to change the priority of the `io-pkt` subsystems.

qos_verbose=*X*

The level of verbosity of the output related to connection management by the QoS. The default is 0; the bigger the number, the more verbosity. This option is used for diagnosis.

res_retries=*X*

The number of retries the Ethernet resolver will perform during an attempted node resolution before giving up. The default is 2.

res_ticks=*X*

The number of periodic ticks before the Ethernet resolver retransmits a node resolution request. The default is 1.

resolve=*resolver*[:*resolver_parameter*]

Add to the resolver list for mountpoints that follow.

The following values for *resolver* are built into the network manager:

- `en_ionet` — broadcast requests for name resolution on the LAN (similar to the TCP/IP ARP protocol). This is the default.
- `dns` — Take the node name, add a dot (`.`) followed by the node domain, and send the result to the TCP/IP *gethostbyname()* function.
- `file` — The *resolver_parameter* is is the name of the file to use; the default is `/etc/qnet_hosts`. The format of the file is as follows:

```
# This is a comment line
host.domain      addr1[,addr2]
...
```

The *host.domain* represents a Qnet fully qualified domain name (FQDN). The *addr1* and optional *addr2* are the interface addresses for the FQDN. For `bind=en`, the format of an address is *xx:xx:xx:xx:xx:xx* (the MAC address).

If you specify something else, Qnet attempts to load `nr-`*resolver*`.so`. The default name resolver is `en_ionet`. For queries how to create `nr-`*resolver*`.so`, please contact Technical Support.

Copyright © 2014, QNX Software Systems Limited

The following combinations aren't supported:

- bind=en and resolve=dns
- bind=ip and resolve=file

ret_drvr_rx_buf=*X*

How L4 should handle the received (*rxd*) packets:

- 0 — hold onto multiple *rxd* packets during reassembly (the default). This results in the best performance.
- 1 — make a copy of the data in the packets and immediately return the packet buffers to the driver. This is useful when there's a limited number of packet buffers provided by the driver.

slow_mode=*X*

The number of ticks after which to forget about the slow transmit mode (i.e. tightly windowed) for a node. The default is 1200, giving 240 seconds or four minutes. The value 0 is special; it disables slow mode entirely.

tx_retries=*X*

The number of times Qnet should retry a transmission before giving up. The default is 25.

tx_ticks=*X*

The number of periodic ticks before L4 retransmits a transmission request. The default is 1.

vtag=*tag_number*

Cause Qnet to insert a four-byte `vlan` tag into the packet. The *tag_number* must be greater than zero. If you use this option, Qnet accepts only packets that have a tag value that matches the given *tag_number*. If the driver being used doesn't support 1518-byte packets, you must also use the Qnet mtu_en=1496 option.

To use Qnet over a VLAN interface, you must first set the parent interface to be “up” to enable the interface:

```
ifconfig interface_name up
```


Copyright © 2014, QNX Software Systems Limited

You then need to create the VLAN interface and bind it to the parent interface in one step:

```
ifconfig vlan0 create vlan tag_number vlanif interface_name up
```

This VLAN interface can exist before Qnet is mounted, or you can dynamically create it after Qnet is started. If you need to change the VLAN parent interface, you must destroy and recreate the VLAN interface.

Description:

The `lsm-qnet.so` is the lightweight version of the manager that implements native QNX Neutrino networking, Transparent Distributed Processing (TDP).

The QoS software layer implements the node-to-node session protocol and handles the selection of transmission media.

L4 is a software layer beneath the QoS that implements an ISO level-4 style transport to provide guaranteed, non-duplicate delivery of data, using the driver layer below it.

You can't `umount` Qnet, but you can create an `io-pkt` producer module that supports unmounting.

If you specify two or more `resolve=` options in a series, the resolvers form a list of lookups for the directory specified in the subsequent mount= options.

A list of resolvers is terminated by a mount= option. Any resolve= options placed *after* a mount= option form a *new* list — they don't add to the previous list.

For example, the following line:

```
resolve=a,resolve=b,mount=x,mount=y,resolve=c,mount=z
```

specifies that:

- mount=x has resolvers `a` and `b`
- mount=y also has resolvers `a` and `b`
- mount=z has *only* resolver `c`

Once Qnet has a domain, you can't set Qnet to not use a domain; you can only change the domain.

Examples:

You can start Qnet in two ways: either you start it at the same time as `io-pkt`, or afterward with the `mount` utility.

Copyright © 2014, QNX Software Systems Limited

Start a network driver, Qnet, and TCP/IP at once, with Qnet using the default DIX blue book ethernet packet type with no CRC checking, and 1024 descriptors for maximum performance:

```
io-pkt -d speedo transmit=1024,receive=1024 -p qnet -p tcpip
```

Start `io-pkt`, and then use the `mount` command to start the driver and Qnet in sequence, using the actual file names of the shared libraries:

```
io-pkt
mount -T io-pkt -o transmit=1024,receive=1024 devnp-speedo.so
mount -T io-pkt lsm-qnet.so
```

These shared libraries (with the `.so` extension) are actually located in `/lib/dll`. The `mount` utility automatically looks for them there. If you wish, you can give the full pathname to the `mount` utility.

The following example shows how you start everything at once using IP packet encapsulation instead of DIX blue book:

```
io-pkt -d speedo transmit=1024,receive=1024 -p qnet bind=ip,resolve=dns -p tcpip
```

See *mount* (p. 1312) and *io-pkt* (p. 1007) for more information.

Files:

`/dev/io-net`

The directory where, by default, protocol modules and legacy `io-net` drivers add entries. For more information, the documentation for *io-pkt* (p. 1007).

`/etc/system/config/useqnet`

If this file exists, your system is using the default startup files, and `io-pkt` is running when your system starts up, the system automatically loads `lsm-qnet.so`.

`/proc/qnetstats`

An entry that Qnet places in the `/proc` filesystem. If you open this name and read from it, the Qnet resource manager code responds with the current statistics for Qnet.

Caveats:

Qnet doesn't fully support communication between a big-endian machine and a little-endian machine. However, it does work between machines of different processor types (e.g., ARMLE-v7, x86) that are of the same endian-ness. If you require cross-endian networking with Qnet, please contact your sales representative.

Copyright © 2014, QNX Software Systems Limited

lsm-slip.so

Loadable stack module that supports Serial Line IP (SLIP) network interface

Syntax:

```
io-pkt-v4-hc ... -p /lib/dll/lsm-slip.so ....
```

or

```
mount -T io-pkt /lib/dll/lsm-slip.so
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `lsm-slip.so` module supports the SL interface, which allows asynchronous serial lines to be used as IPv4 network interfaces using the SLIP protocol.

To use the SL interface:

1. Load `lsm-slip.so` into *`io-pkt-v4-hc`* (p. 1007).
2. Create the interface, using the *`ifconfig create`* (p. 957) subcommand.
3. Use *`slattach`* (p. 1772) to assign a TTY line to the interface.

Once the interface is attached, you can configure network source and destination addresses and other parameters via `ifconfig`.

The `lsm-slip.so` interface can use Van Jacobson TCP header compression and ICMP filtering. The following flags to `ifconfig` control these properties of a SLIP link:

`link0`

Turn on Van Jacobson header compression.

`-link0`

Turn off header compression (the default).

`link1`

Don't pass through ICMP packets.

`-link1`

Copyright © 2014, QNX Software Systems Limited

Pass through ICMP packets (the default).

link2

If a packet with a compressed header is received, automatically enable compression of outgoing packets (the default).

-link2

Don't auto-enable compression.

For more information, see:

- J. Romkey, *A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP, RFC, 1055*, June 1988.
- Van Jacobson, *Compressing TCP/IP Headers for Low-Speed Serial Links, RFC, 1144*, February 1990.

SLIP can transmit only IPv4 packets between preconfigured hosts on an asynchronous serial link. It has no provision for address negotiation, carriage of additional protocols (e.g., XNS, AppleTalk, DECNET) and isn't designed for synchronous serial links. This is why SLIP has been superseded by the Point-to-Point Protocol (PPP), which does all of those things, and much more.

Diagnostics:

sl%d: af%d not supported

The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

lwresd

Lightweight resolver daemon

Syntax:

```
lwresd [-c config-file] [-C config-file] [-d debug-level] [-f]
       [-g] [-i pid-file] [-m flag] [-n num_cpus] [-P port]
       [-p port] [-s] [-t directory] [-u user] [-v]
       [-4] [-6]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?lwresd+8+NetBSD-5.0* in the NetBSD documentation.

Description:

The `lwresd` utility is the daemon providing name-lookup services to clients that use the BIND 9 lightweight resolver library. It's essentially a stripped-down, caching-only name server that answers queries using the BIND 9 lightweight resolver protocol rather than the DNS protocol. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?lwresd+8+NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

Chapter 14
M

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`mkxfs`	*`mkefs`* (p. 1209), *`mketfs`* (p. 1219), *`mkifs`* (p. 1241), *`mkfatfsimg`* (p. 1228), *`mkqnx6fsimg`* (p. 1279), *`mkrcfsimg`* (p. 1296)
`mmcsdpub`	*Device Publishers Developer's Guide*

This chapter describes the utilities, etc. whose names start with “M”.

Copyright © 2014, QNX Software Systems Limited

m4

Macro processor (GNU)

Syntax:

`m4 [`*`options`*`] [`*`files`*`]`

Runs on:

Linux, Microsoft Windows

Options:

See *http://www.gnu.org/software/m4/manual/*.

Description:

The `m4` utility is a macro processor, in the sense that it copies its input to the output, expanding macros. For detailed documentation, see *http://www.gnu.org/software/m4/manual/*.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

Copyright © 2014, QNX Software Systems Limited

/usr/share/misc/magic

Magic-number file for the `file` command (UNIX)

Name:

`/usr/share/misc/magic`

Description:

The *`file`* (p. 746) command identifies the type of a file using, among other tests, a test for whether the file begins with a certain *magic number*. The file `/usr/share/misc/magic` specifies what magic numbers are to be tested for, what message to print if a particular magic number is found, and additional information to extract from the file.

Each line of the file specifies a test to be performed. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, or 4-byte numeric value or a string. If the test succeeds, a message is printed. The line consists of the following fields:

offset

A number specifying the offset, in bytes, into the file of the data which is to be tested.

type

The type of the data to be tested. The possible values are:

`byte`

A one-byte value.

`short`

A two-byte value (on most systems) in this machine's native byte order.

`long`

A four-byte value (on most systems) in this machine's native byte order.

`string`

A string of bytes.

`date`

Copyright © 2014, QNX Software Systems Limited

A four-byte value interpreted as a UNIX date.

beshort

A two-byte value (on most systems) in big-endian byte order.

belong

A four-byte value (on most systems) in big-endian byte order.

bedate

A four-byte value (on most systems) in big-endian byte order, interpreted as a UNIX date.

leshort

A two-byte value (on most systems) in little-endian byte order.

lelong

A four-byte value (on most systems) in little-endian byte order.

ledate

A four-byte value (on most systems) in little-endian byte order, interpreted as a UNIX date.

The numeric types may optionally be followed by & and a numeric value, to specify that the value is to be ANDed with the numeric value before any comparisons are done. Prepending a u to the type indicates that ordered comparisons should be unsigned.

test

The value to be compared with the value from the file. If the *type* is numeric, this value is specified in C form; if it's a *string*, it's specified as a C string with the usual escapes permitted (e.g. \n for newline).

Numeric values may be preceded by a character indicating the operation to be performed:

- =, to specify that the value from the file must equal the specified value
- <, to specify that the value from the file must be less than the specified value
- >, to specify that the value from the file must be greater than the specified value
- &, to specify that the value from the file must have set all of the bits that are set in the specified value

Copyright © 2014, QNX Software Systems Limited

- ^, to specify that the value from the file must have clear any of the bits that are set in the specified value
- x, to specify that any value matches.

If the character is omitted, it's assumed to be =.

Numeric values are specified in C form; e.g. `13` is decimal, `013` is octal, and `0x13` is hexadecimal.

For string values, the byte string from the file must match the specified byte string. The operators =, < and > (but not &) can be applied to strings. The length used for matching is that of the string argument in the `/usr/share/misc/magic` file. This means that a line can match any string, and then presumably print that string, by doing `>\0` (because all strings are greater than the null string).

message

The message to be printed if the comparison succeeds. If the string contains a *printf()* format specification, the value from the file (with any specified masking performed) is printed using the *message* as the format string.

Some file formats contain additional information that's to be printed along with the file type. A line that begins with the character > indicates additional tests and messages to be printed. The number of > characters on the line indicates the level of the test; a line with no > at the beginning is considered to be at level 0. Each line at level *n+1* is under the control of the line at level *n* most closely preceding it in the magic file.

If the test on a line at level *n* succeeds, the tests specified in all the subsequent lines at level *n+1* are performed, and the messages printed if the tests succeed. The next line at level *n* terminates this.

If the first character following the last > is a `(`, then the string after the parenthesis is interpreted as an indirect offset. That means that the number after the parenthesis is used as an offset in the file. The value at that offset is read, and is used again as an offset in the file. Indirect offsets are of the form: `((`*x*`[.[bsl]][+-][`*y*`])`. The value of *x* is used as an offset in the file. A byte, short or long is read at that offset depending on the `[bsl]` type specifier. To that number the value of *y* is added and the result is used as an offset in the file. The default type if one isn't specified is long.

Caveats:

The formats `long`, `belong`, `lelong`, `short`, `beshort`, `leshort`, `date`, `bedate`, and `ledate` are system-dependent; perhaps they should be specified as a number of bytes (2B, 4B, etc), since the files being recognized typically come from a system on which the lengths are invariant.

There's (currently) no support for specified-endian data to be used in indirect offsets.

Copyright © 2014, QNX Software Systems Limited

make

Maintain, update, and regenerate groups of programs (POSIX)

Syntax:

```
make [options] [macro=name] [target]
```

Runs on:

Linux, Microsoft Windows

Options:

-C *directory*

Change to the given *directory* before doing anything.

-d

Display debugging information.

-e

Cause environment variables to override macro assignments within *makefiles*.

-f *makefile*

Use the description file *makefile*. If the pathname is the dash character (-), standard input is used. The default file is `makefile` or `Makefile`.

If there are multiple instances of this option, they're processed in the order specified.

-h

Display help information.

-i

Ignore error codes returned by commands. This is equivalent to the special `.IGNORE:` target.

-I *directory*

Search *directory* for included makefiles.

-j [*N*]

Allow *N* jobs at once. The default is infinite.

Copyright © 2014, QNX Software Systems Limited

-k

If an unignored error occurs while executing the commands to bring a target up-to-date, abandon work on the current target, but continue with targets that don't depend on the current target. This is the opposite of -S. If both -k and -S are specified, the last one specified is used.

-l [*N*]

Don't start multiple jobs unless the load is below *N*. If *N* isn't specified, this option removes a previous load limit.

-n

No-execution mode. Print commands, but don't execute them. However, lines preceded by a plus sign (+) prefix or containing the string `$(make)` *are* executed. Lines beginning with an @ are printed.

-o *file*

Consider *file* to be very old and don't remake it.

-p

Write to standard output the complete set of macro definitions and target descriptions.

-q

Question mode. The `make` utility returns a zero or nonzero status code, depending on whether or not the target file is up-to-date. Targets aren't updated if this option is specified.

-r

Clear the suffix list and don't use the default inference rules.

-s

Be silent. Don't print command lines before executing them. This is equivalent to the special `.SILENT:` target.

-S

Undo the effect of the -k option. Stop processing when a nonzero exit status is returned by a command. If both -k and -S are specified, the last one specified is used. This option overrides the presence of the k flag in the ***MAKEFLAGS*** environment variable.

-t

Copyright © 2014, QNX Software Systems Limited

Touch the target files, changing only the file date, rather than perform the rules to reconstruct them.

-v

Print the version number of make, and then exit.

-w

Print the current directory.

-W *file*

Consider *file* to be infinitely new.

--no-print-directory

Turn off -w, even if it was turned on implicitly.

--warn-undefined-variables

Warn when an undefined variable is referenced.

target_name

The target to bring up-to-date. If no target is specified, make uses the first target defined in the first *makefile* encountered.

macro=name

Macro definition. This definition remains fixed for the make invocation. It overrides any regular definitions for the specified macro within the *makefiles* and from the environment. This definition is inherited by subordinate make sessions but acts as an environment variable for these. That is, depending on the -e setting, it may be overridden by a *makefile* definition.

Description:

The make utility is used to maintain current versions of files by resolving time dependencies. If the file that make is examining (i.e. the target) is older than the file(s) on which it depends (i.e. the prerequisites), then make executes a list of shell commands associated with the target to create or update the target.

To control what make does, the utility refers to specifications (rules) consisting of a target, prerequisites, and a list of shell commands to be executed when a prerequisite is newer than the target. To simplify maintenance of programs, make has a collection of default macros and inference rules allowing it to identify the proper procedures required to update targets based on minimal information. (If the -r option is given, the builtin rules aren't used.)

Copyright © 2014, QNX Software Systems Limited

For example, to compile `myprog.c` and produce the executable program `myprog`:

```
make myprog
```

This works even in the absence of a makefile. The alternative of using the `qcc` utility produces an executable named `a.out` by default. Using `make` for compiling single-source executables is a convenient shortcut.

When cross compiling, you can't rely on the default rules; you need to specify the target.

The `make` utility attempts to perform the actions required to ensure that the specified target(s) are up-to-date. A target is considered out-of-date if it's older than any of its prerequisites. The `make` utility treats all prerequisites as targets themselves and recursively ensures that they're up-to-date. The utility examines the modified times of files to determine whether they're out-of-date.

After all the prerequisites of a target are ensured to be up-to-date, and if the target is out-of-date, the shell commands associated with the target entry are executed. If no shell commands are listed for the target, it's treated as up-to-date.

For more information on `make` and writing makefiles, see:

- the GNU website at *http://www.gnu.org/*
- Andrew Oram and Steve Talbott, *Managing Projects with make*, O'Reilly and Associates, 1991

Examples:

```
make myfile.o
```

Typing this in the absence of a makefile (or with a makefile that doesn't mention the `myfile.o` file) uses the builtin suffixes list and inference rules to determine the name of the source file and, if necessary, to run the proper command to create or rebuild the `myfile.o` file.

Suppose you have the source files `myfile1.c`, `myfile2.c`, and `myfile3.c`. The first two files include the headers `<hdr1.h>` and `<hdr2.h>`, and these files are all linked together to produce the program `myprog`. Here's what a simple makefile describing this could look like:

```
#  Samplemakefile1
myprog:  myfile1.o myfile2.o myfile3.o
myfile1.o:  hdr1.h hdr2.h
myfile2.o:  hdr1.h hdr2.h
myfile3.o:
```

Copyright © 2014, QNX Software Systems Limited

To compile and link `myprog`:

```
make myprog
```

Or since `myprog` is the first target:

```
make
```

To see what commands need to be executed without actually executing them:

```
make -n
```

Using macros, the `myprog` makefile could be simplified to:

```
#  Samplemakefile2

OBJS=myfile1.o myfile2.o myfile3.o
HDRS=hdr1.h hdr2.h

myprog:  $(OBJS)

myfile1.o:  $(HDRS)

myfile2.o:  $(HDRS)
```

This makefile is functionally identical to the previous example. It isn't significantly better, just slightly more generalized.

We can further simplify the makefile by using the builtin inference rules and macros, as follows:

```
#  Samplemakefile3

OBJS=myfile1.o myfile2.o myfile3.o
HDRS=hdr1.h hdr2.h

myprog:  $(OBJS)

myfile1.o myfile2.o: $(HDRS)
```

As you can see, this makefile is short and to-the-point. Again, this is functionally equivalent to the previous examples.

Using this makefile, you can customize the compilation from the `make` command line by setting the appropriate macros:

```
make CFLAGS="-DHITHERE"
```

which defines the symbol `HITHERE`.

You can also direct the linker to produce a linkmap:

```
make LDFLAGS=-M
```

Of course, any of these macro assignments could be “hard-coded” in the makefile, but it's often convenient to override the defaults from the command line for special needs.

Environment variables:

MAKEFLAGS

Copyright © 2014, QNX Software Systems Limited

This environment variable, if it exists, is read by the make utility, and is treated as a set of option characters to be used as the default options. Command-line options to make have precedence over these options.

After make has started and the ***MAKEFLAGS*** variable has been read, all of the options except -d, -f, and -p are added to the *MAKEFLAGS* macro. The *MAKEFLAGS* macro is passed into the environment as an environment variable for all child processes.

SHELL

The value of the ***SHELL*** environment variable is always ignored. All other environment variables are used as macros.

Exit status:

When the -q option has been specified:

0

Successful.

1

The target wasn't up-to-date.

>1

An error occurred.

When the -q option hasn't been specified:

0

Successful.

>0

An error occurred.

Contributing author:

GNU

Caveats:

In makefiles, specify Windows pathnames using one of the following methods:

- Use a forward slash (/) as a path separator.
- Use a backslash (\) escape character before a backslash path separator.
- Enclose the entire path in quotation marks.

Copyright © 2014, QNX Software Systems Limited

mcd

Media Content Detector utility

Syntax:

```
mcd options* config_file &
```

Runs on:

QNX Neutrino

Targets:

ARM, x86

Options:

-D

Set the name of the mediastore list directory; the default is `.devices`.

-E

Set the name of the mediastore eject file; the default is `.eject`.

-I

Set the name of the mediastore insert file; the default is `.insert`.

-L

Run the MCD in local mode, even when Qnet is running. See "*Local mode* (p. 1173)" below.

-n

Set the subsystem mountpoint; the default is `/dev/mcd`.

-v

Increase the verbosity of messages written to `sloginfo`, from 0 to 7.

-V

Print output messages to console, as well as `sloginfo`.

config_file

The pathname to a required configuration file; see "*Configuring the MCD* (p. 1175)" below.

Copyright © 2014, QNX Software Systems Limited

Local mode

The local mode option (-L) may permit simpler configuration when stand-alone systems are used, as describe here.

Use local mode if a system using MCD is to be run as a stand-alone with respect to any services that use the MCD, and Qnet is running on the system (for example, for debugging).

When the MCD runs in local mode, the MCD:

- does *not* add the network prefix to paths it generates
- expects all paths it receives to already have the network prefix

Limitations

When the local mode option is used:

- The MCD configuration file may *not* contain network-qualified devices.
- Off-node processes can *not* open rule entries (for example, `DISC_INSERTED`), so they are unable to read out a local-only path.
- To prevent any ambiguity between local and remote device names, off-node processes can *not* open control entries (for example, `.insert` or `.eject`).
- Path names handed out to *read()* are *local* only — even if Qnet is running.

Description:

The `mcd` utility (media content detector, or MCD) monitors device and mediastore insertions and removals, and the presence of specified media content.

This utility page contains:

Overview

Copyright © 2014, QNX Software Systems Limited

The MCD is a stand-alone utility for detecting devices, mediastores and specified media content. It is positioned between storage and USB device drivers, and any client application that needs to be informed of a device or mediastore activity, or of the presence of specified media types.

The MCD design separates the definition of actions conducted by the system from the implementation of these actions so that the actions can be easily edited or updated without changes to code.

MCD rules

The MCD provides a binary decision tree framework, applying rules and branching between rules according to match/fail results. These rules are used for detecting device and mediastore insertions and removals, and for classifying their content. They are specified in the MCD *configuration file* (p. 1175), and implemented with *user callouts* (p. 1180).

You can write rules for the MCD instructing it to monitor the presence or absence of any device, mediastore or file, as shown by the three examples below:

Monitor a mediastore

The rule below tells the MCD to monitor the physical insertion or removal of a CD-ROM mediastore on a device (the hardware) at the location `/dev/cd*`.

```
[/dev/cd*]
Callout=CD_MEDIA_IOBLK
```

Monitor namespace changes

The rule below monitors changes (mount or unmount) of a device or mediastore (such as a USB storage device) on the system. These sorts of changes usually indicate the physical insertion or removal of the device (the hardware) and its filesystem mounting or unmounting.

```
[/fs/usb*]
Callout=PATH_MEDIA_PROCMGR
```

Monitor the presence of files

The rule below polls the contents of `/directory` looking for files or directories that have been created in or removed from the directory being monitored. For example, typing `touch /directory/file` from a shell satisfies this rule, though no physical device has been inserted or removed.

```
[/directory/*]
Callout=PATH_MEDIA_SCAN
```

The MCD can be used as a framework from which to build an independent content detection system.

MCD server

Copyright © 2014, QNX Software Systems Limited

The MCD server:

- monitors a user-specified set of devices and their mediastores
- attempts to determine the category of media on mediastores (for example, an audio CD or a movie on a DVD)
- notifies clients of the presence of the media it has detected

In this documentation:

- *mediastore* is never a specific mediastore, but always *any mediastore of the specified type*: a CD and not, for example, the CD *I'm Your Man* by Leonard Cohen.
- an unformatted CD or USB stick is considered a device, because in its current state it cannot have any media content
- a USB stick with, for example, two partitions is one device with two mediastores

Operational flow

This section describes the MCD's operational flow at startup, and on detection of a new device, mediastore or file.

Startup

At startup, the MCD server proceeds as follows:

1. Read the configuration file.
2. Create a single, dedicated thread to provide the resource manager interface.
3. For each mediastore listed in the configuration file:
   a. Create a dedicated, device detection thread.
   b. Run the notification routine in its own thread.

Multiple detection threads, each for different mediastores, may be running concurrently.

Device or insertion

On detection of a device or mediastore insertion, or of the presence of a file of interest, the MCD proceeds as follows:

- Create a new thread to process the content detection rules for that device, mediastore or file.
- When the rule processing is complete, terminate the thread.

Configuring the MCD

Copyright © 2014, QNX Software Systems Limited

The operation of the MCD is controlled by a configuration file. This file consists of named sections, each section defined by a name enclosed in square brackets: `[ ]`, followed by parameter lines with the form `key = value`. These parameters apply only to the section in which they appear.

The sample MCD configuration file `2phase.cfg` is delivered with the MCD; the sample configuration file `mcd.conf` provides examples for use with the Aviage Multimedia suite (MME).

The MCD ignores blank lines and any leading or trailing white spaces. It treats lines beginning with a "#" or a ";" character as comments and ignores them as well.

Configuration file sections

A section of the MCD configuration file can be one of:

- an entity (device, mediastore or file) description — the section name starts with a "/" character (i.e. `/dev/cd0`). See "*Entity descriptions* (p. 1176)" below.
- a media content rule — any name without a leading "/". See "*Media content rules* (p. 1177)" below.

The example below presents a description: `[/dev/cd0]` and a rule: `[DVD_AUDIO]`:

```
[/dev/cd0]
Callout     =   CD_MEDIA_IOBLK
Argument    =   1000,2000
Priority    =   11,9
Start Rule  =   DVD_OR_CD

[DVD_AUDIO]
Callout     =   FNAME_MATCH
Argument    =   /AUDIO_TS/AUDIO_TS.IFO
Match Rule  =   DVD_VIDEO
Fail Rule   =   DVD_VIDEO
```

Entity descriptions

For entity (device, mediastore or file) description sections, the section name is the entity the MCD monitors. This name can be a single name, such as `/media/drive`, or a wildcard pattern, such as `/dev/umass*`. If the section name is a wildcard pattern, the event notification routine defined by the `Callout=` for the section must be capable of handling every entity that matches the wildcard pattern.

Parameters

Configuration parameters are used differently according to the type of section (mediastore description, or content rule) in which they are used.

Parameters in an entity section of the configuration file are used as follows:

`Callout=`

Copyright © 2014, QNX Software Systems Limited

The notification routine that the MCD runs when it detects the entity defined in the section name. Each notification routine is run in its own thread, allowing it to either block or poll as necessary.

If you provide no `Callout=` routine, you should handle device, file or mediastore transitions externally with the notification provided through the *resource manager interface* (p. 1179).

`Argument=`

An optional argument to pass to the notification routine defined by the `Callout=` parameter.

`Priority=`

The priorities at which to run, if applicable, for the entity defined in the current section:

- the content detection thread
- the notification thread

`Start Rule=`

The root of the decision tree executed to determine media content type; that is, the first rule to apply to each entity following insertion notification.

`Stop Rule=`

The root of the callouts to execute when the entity is removed.

Media content rules

For media content rule sections of the `mcd` configuration file, the section name is the name of the rule.

Parameters

Parameters in a media content rule section of the configuration file are used as follows:

`Callout=`

The notification routine that the MCD runs when it detects an entity matching the content rule in the section name. See "*Notification routine* (p. 1178)" below.

`Argument=`

An optional argument to pass to the notification routine defined by the `Callout=` parameter.

`Match Rule=`

Copyright © 2014, QNX Software Systems Limited

The branch of the decision tree to execute when media content matches the content rule.

`Fail Rule=`

The branch of the callouts to execute when no media content matches the content rule.

A rule runs at the priority given in the *entity section* (p. 1176) that starts the rule chain.

Notification routine

The notification routine that runs when a mediastore matches a rule produces a match/fail result to indicate whether or not the media on the mediastore satisfies the routine's particular requirements. Based on the result produced by the rule, the MCD takes a branch to another rule, as specified by the `Match Rule=` or `Fail Rule=` parameters in the current section.

If no associated branch rule is provided for a rule's result, the MCD considers the rule to be terminal and content detection complete.

If a rule contains no `Callout=` parameter, the MCD assumes that the rule either matches or fails, based on the presence of an associated branch rule. When debugging, you can use this characteristic to disable a test and always make a branch to the next rule.

Using the MCD as a filesystem automounter

The MCD can be used as a filesystem automounter by creating a set of two-phase rules in the MCD configuration file. Two-phase rules are implemented in the MCD as follows:

First-level entries

- The first level of entries in the MCD configuration file refers, *not* to mediastores, but to devices that can be monitored using the provided MCD callouts.
- The `Start Rule=` parameter in a first level entry points to the `MOUNT_FSYS` callout, a built-in routine that can mount filesystems based on simple mediastore criteria.
- Mounting a filesystem uses the triggers insertion notification of the second level of entries (for mediastores) via the `PATH_MEDIA_PROCMGR` built-in routine.

Second-level entries

- When they are triggered, the MCD configuration file's second-level entries perform the content detection algorithms on the mediastores at the filesystem mountpoint.

Copyright © 2014, QNX Software Systems Limited

For an example of how to use the MCD as a filesystem automounter, see "*Two-phase filesystem mount example* (p. 1194)" in the "Examples" section below.

The mcd resource manager interface

The MCD server presents a standard QNX Neutrino resource manager (filesystem-like) interface. The default top-level directory is `/dev/mcd`; it includes:

- a set of `S_IFNAM/name-special` (*`.insert` and `.eject`* (p. 1179)) files, which the MCD uses to provide a *client API* (p. 1187) to the system
- a *`.devices`* (p. 1179) directory with an entry for each entity known to the system

To change the top-level directory, use the -n command-line option.

`.insert` and `.eject` files

The `.insert` and `.eject` files are write-only files in `/dev/mcd`. External programs can trigger the MCD's content detection process on the insertion or ejection of an entity by writing to the appropriate file the pathname of the entity that has been inserted or ejected; for example `/dev/cd1` .

If the hardware for the device with an ejected mediastore doesn't support removal notification, then the MCD treats a subsequent insertion notification as an implicit ejection event.

`.devices` directory

The `.devices` directory in `/dev/mcd` contains an entry for each entity (device, mediastore and monitored file) known to the system. Each entity is represented by a `S_IFCHR/char-special` file in this directory. These files hold information about the entity in fields as follows:

- *st_mtime* — the time of the last state change for the entity.
- *st_ino* — the sequence number of the insertion. See "*Sequence number* (p. 1180)" below.

An entry appears in `.devices` directory only for entities that have been inserted at least once. Since entity sections can be wildcards, a full list of potential entity matches can not be known in advance. The MCD can only know about an entity after it has been inserted. Thus, if a client application tries to *stat()* a particular device and fails with `ENOENT`, it should treat this failure as though the *st_ino* field is 0 (i.e., device ejected).

The "*Client API* (p. 1187)" section below includes an example that illustrates how to use this interface.

Copyright © 2014, QNX Software Systems Limited

Sequence number

The sequence number stored in *st_ino* for any entity (device, mediastore or file) can be either zero or non-zero. A value of zero means that the entity is not present in the system. A non-zero value means that the entity is present in the system.

At each re-insertion of the entity, the MCD increases the sequence number for that entity. Thus, for example, for a mediastore the values of *st_ino* might be in sequence: 1 (first insertion), 0 (removal), 3 (re-insertion), 0 (removal), 5 (re-insertion).

A client application can use the incrementing value of *st_ino* at each state change to check that it is up to date with, for example, a mediastore's activity after a series of rapid insertions and removals. For more information, see "*Stale rules* (p. 1188)" below.

Example: Filesystem hierarchy

Below is a sample filesystem hierarchy:

```
$ ls -al /dev/mcd
dr-xr-xr-x  1 root   root  11 Aug 02 19:46 .
n-w--w--w-  1 root   root   0 Aug 02 19:46 .eject
n-w--w--w-  1 root   root   0 Aug 02 19:46 .insert
nr--r--r--  1 root   root   0 Aug 02 19:46 CDDA_OR_DTS
nr--r--r--  1 root   root   0 Aug 02 19:46 CD_AUDIO
nr--r--r--  1 root   root   0 Aug 02 19:46 DVD_AUDIO
nr--r--r--  1 root   root   0 Aug 02 19:46 DVD_OR_CD
nr--r--r--  1 root   root   0 Aug 02 19:46 DVD_VIDEO
nr--r--r--  1 root   root   0 Aug 02 19:46 MIXED_AV
nr--r--r--  1 root   root   0 Aug 02 19:46 SVIDEO_CD
nr--r--r--  1 root   root   0 Aug 02 19:46 VIDEO_CD
```

Read-only entries for rules

The top-level `/dev/mcd` directory contains read-only entries for each rule defined in the configuration file.

Client applications can read from here the name of the device that satisfied a particular rule, with the read blocking until a device with content matching that rule is available. A non-blocking select and notify mechanism is also available to allow the client to wait on multiple rules, or to perform other work until a rule is triggered. Following the notification, the client application can read the rule to determine the device.

Callout templates

The MCD server provides a framework from which you can build a content detection system. Callout routines provide all the specific functionality in such a content detection system.

The MCD includes some common routines available for use where required in a static-linked library bound into the server. The MCD also supports extension routines provided by third-parties in DLLs dynamically linked at runtime. Thus, the system is extensible: if you require a new, unsupported detection test, you can implement it outside the server framework and ship it as a separate library.

In the content detection system configuration file, all `Callout=` items refer to a callout. These callouts are identified as internal or external by their names:

Copyright © 2014, QNX Software Systems Limited

- If the function name contains an “at” (@) character (for example, `my func@mylib.so`), it refers to an external function in the named DLL, which is resolved at runtime.
- If the function name doesn't contain an @ character, the callout is an internal built-in routine.

Extension modules must include the MCD header file `<sys/mcd.h>` for appropriate manifests and type definitions.

Insertion and ejection notification

The prototype for media insertion notification callouts is:

```
void mcd_notify( char *iomgr[2], char *device, void *arg );
```

The MCD creates this routine in a dedicated thread, that should continually monitor the device. This thread should not return, *except* in the event of a serious error. If the thread encounters a serious error, it should set *errno* appropriately and return. On the return of an entity detection thread, the MCD will:

- log the error
- stop monitoring the entity whose monitoring thread encountered the error
- continue monitoring other entity detection threads

Arguments

iomgr

An array containing the names of the interface entries where insertion and ejection events are reported. In a default system iomgr[0] = "/dev/mcd/.insert" and iomgr[1] = "/dev/mcd/.eject", but the actual strings will reflect any command-line overrides.

device

A pointer to the name of the device, mediastore or file to monitor.

device may be a wildcard, requiring the routine to monitor a group of devices, files or mediastores. When a device event occurs, the routine should write to the appropriate *iomgr[]* path the name of the specific device, mediastore or file that is affected by the event.

arg

A pointer to a routine-specific argument. This routine-specific argument is provided as the `Argument=` parameter of the relevant device entry in the configuration file.

Built-in notification routines

Copyright © 2014, QNX Software Systems Limited

The MCD's built-in media notification routines include:

CD_MEDIA_IOBLK

This routine interfaces the system to CD and DVD drives managed by the `devb/io-blk` filesystem. It exploits the filesystem feature of automatically invalidating open file descriptors upon media change: it opens the device block-special file and periodically polls it, treating any EBADF result as a transition indicator. It distinguishes between insertion and ejection by examining the advertised size of the device.

The `Argument=` option sets the polling periods (in milliseconds) at which to probe the file descriptor when the mediastore missing, and when it is present. The default for this argument is `"1000,2000"`, which corresponds to a one-second interval when the mediastore is missing, and a two-second interval when the mediastore is present.

USB_MEDIA_ENUM

Targets running QNX Neutrino 6.3.*n* releases only. This routine interfaces the system to the `umass-enum` utility (which is replaced by `enum-usb` in QNX Neutrino 6.4.0). It translates USB state change events from their native format into the insertion and ejection events expected by the USB enumeration server named by the `Argument=` option.

The `Argument=` option sets the name of the USB enumeration server to connect to (typically `"/dev/umass-enum"`).

PATH_MEDIA_PROCMGR

This routine connects the system to any device that dynamically attaches a resource manager pathname when the resource manager is present, and detaches the pathname when the resource manager is absent. It uses the *procmgr_event()* facility (available from QNX Neutrino 6.3.0 SP2 onwards) to receive notification of any changes to the global pathname space, then scans for the addition and/or removal of any device mountpoints matching the pattern defined by the callout name.

The `Argument=` option sets the name of the special directory where the OS `pathmgr` maintains mountpoints (typically `"/proc/mount"`).

PATH_MEDIA_SCAN

This routine scans for the presence of filenames that match the pattern defined by the callout name. It is similar to the `PATH_MEDIA_PROCMGR` routine, but since the creation and deletion of files doesn't trigger any filesystem events, this routine operates by scanning the specified directory at regular intervals.

Copyright © 2014, QNX Software Systems Limited

The PATH_MEDIA_SCAN cause the MCD to behave differently, based on the presence or absence of a trailing "/" character at the end of the pathname, as follows:

- "/" present — send one notification when the pathname exists as a directory (including as the root directory of a mounted filesystem)
- "/" absent — send a notification for every matching filename pattern in the named directory

The Argument= option sets the poll period, in milliseconds, for scanning the directory.

Media content determination

The prototype for content detection rule callouts is:

```
int mcd_content( char *device, void *arg );
```

Arguments

device

The name of the raw device or mediastore

arg

A pointer to a routine-specific argument. This routine-specific argument is provided as the Argument= parameter of the relevant rule entry in the configuration file.

Returns

This routine returns values as follows:

- MCD_RULE_MATCHED — the rule is matched
- MCD_RULE_NO_MATCH — there is no match for the rule
- MCD_RULE_ABORT — there is a serious error; set *errno* and stop the remainder of the detection process for the device

If the routine invoked by the callout requires access to a filesystem on the device, it can use the DCMD_FSYS_MOUNTED_BY devctl to locate the appropriate mountpoint.

Built-in content detection rules

The MCD's built-in content detection routines include:

DVD_OR_CD

Copyright © 2014, QNX Software Systems Limited

This rule determines if a disk mediastore is a DVD rather than a CD (by issuing a `READ DVD STRUCT` command).

It ignores The `Argument=` option. The rule matches only if the media is a DVD.

`CD_AUDIO`

This rule determines if the CD media has any audio content (by issuing a `READ TOC` command). It ignores the `Argument=` option. The rule matches only if the media contains audio tracks.

To facilitate the detection of mixed-mode and enhanced CDs that contain both audio and filesystem components, you can configure the `CD_AUDIO` rule as a non-terminal state; that is, with both the `Match Rule=` branch and the `Fail Rule=` branch provided. With the rule configured this way, after matching audio content, you can continue on with other rules to detect data content.

`BLANK_CD`

This rule determines if the CD media is blank or unrecorded (by issuing a `READ DISK INFORMATION` command). It ignores the `Argument=` option.

The `READ DISK INFORMATION` command and the physical detection of blank disks is only supported by newer CD-RW hardware, and will fail on older CD-ROM hardware. In fact, older hardware may not even detect the insertion of a blank or unrecorded disk.

`FNAME_MATCH`

This rule uses *fnmatch()* to match file pathnames on a device's filesystem. It requires that the inserted mediastore either be a filesystem, or have a filesystem mounted on it, because it is implemented with *access()* probing. The rule will automatically resolve to the filesystem level, if appropriate.

This behavior means that:

- If you are running this rule chain off a device that was a filesystem (i.e. `[/fs/cd*]`), this filesystem is the mountpoint at which the `FNAME_MATCH` rule will search for filenames.
- If you come to the `FNAME_MATCH` rule from a device (i.e. `[/dev/cd0]`), the rule will:

  1. Work out where `/dev/cd0` is mounted (probably `/fs/cd0`).
  2. Search that filesystem for the matching filenames.

Copyright © 2014, QNX Software Systems Limited

The Argument= option is a comma-separated list of pathnames, based from the root of the filesystem. If the MCD finds any of the pathnames in the list on the mediastore, the rule is matched.

FNAME_PATTERN

This rule uses *fnmatch()* to match filename patterns on a mediastore's filesystem. It requires that the inserted mediastore either be a filesystem or have a filesystem mounted on it, because it is implemented with a *nftw()* traversal. The rule will automatically find any such filesystem.

The Argument= option sets a comma-separated list of patterns. If a filename matching any of the listed patterns exists in any directory on the filesystem, the rule is matched.

This option supports several other options that can be embedded in the listed patterns. For example: Argument = depth=2,*.c,*.h". These "embedded" options are:

- basedir= — set the subdirectory at which to start the scan; the default is the root directory of the given entity
- depth= — set the maximum number of subdirectory levels to recurse into, (i.e. the maximum depth from the root); the default is 0, which means no depth limit

By default the scan for a pattern match is the entire target filesystem. You can use the basedir= and depth= options to direct and limit this scan.

MOUNT_FSYS

This rule is used to mount a filesystem onto a specified device, and is used to extend the MCD to operate as an auto-mounter.

The Argument= option sets the filename of a file of mount rules. Since this option is opened and parsed each time the rule is run, you should consider locating this filename on a ramdisk or in /dev/shmem.

This rule is typically used as the Start Rule= of a two-phase configuration, where the resulting mount operation triggers a PATH_MEDIA_PROCMGR action. The rules are processed from the file in order, stopping at the first (*fnmatch()*) match that either succeeds or specifies to skip the device (when the rule has only a pattern and no mount information). In order to select the appropriate filesystem, you can specify multiple rules for a removable device.

The file format is one rule per line, with each line containing fields separated by white spaces. For example:

```
#Device_pattern  Mount_point  Fsys_type  Mount_options
/dev/cd*  /fs/cd%#  udf normv,format=udf,rrip,joliet,iso9660e,iso9660,audio
/dev/umass*t1[124]  /fs/usb%0  dos  fsi=use
```

Copyright © 2014, QNX Software Systems Limited

```
/dev/umass*t[146]  /fs/usb%0  dos
/dev/hd*
```

The rules shown in the example above instruct the MCD to:

- Mount CD with `fs-udf`, specifying the preferred list of formats to try; in order, this list will be UDF, RockRidge, Joliet, then plain ISO9660.
- Ignore HD devices.
- Mount `/dev/cd`*n* as `/fs/cd0`*n*.
- Mount USB partitions as DOS to the first available filesystem mountpoint.

See "`MOUNT_FSYS` *special sequences* (p. 1186)" below for more information about the mountpoint sequences. For more information about UDF, see *`fs-udf.so`* (p. 829).

The *normv* option is required to work around OS namespace race conditions if the filesystem format is not known when trying multiple mounts with different DLLs. It is also useful to prevent a mount if the media is ejected before the rule is scheduled to run.

UNMOUNT_FSYS

This rule unmounts the filesystem from the mediastore specified in the rule name. It ignores the `Argument=` option.

This rule is typically used as the `Stop Rule=` of a `CD_MEDIA_IOBLK` mediastore that uses the `MOUNT_FSYS` action, when a mount would otherwise persist after the mediastore ejection. If the mediastore is presented by a `resmgr` that will exit or be terminated by an external manager (such as USB), then that presentation implicitly unmounts any relevant filesystem without the need for this rule. However, in most instances where a `MOUNT_FSYS` is used, you should also configure a matching `UNMOUNT_FSYS` in order to ensure that the filesystem for an ejected mediastore is duly unmounted.

MOUNT_FSYS special sequences

The `MOUNT_FSYS` rule uses special sequences, as follows:

- `%#` expands to the major device number of the mediastore. For example, with `/fs/cd%#`, `/dev/cd0` will be mounted at `/fs/cd0` and `/dev/cd1` will be mounted at `/fs/cd1`. `%#` does not permit multiple partitions; use `%0`.
- `%0` expands to a sequential, unique number. It is used principally to allow multiple USB partition rules in the `mcd.mnt`, all with the mountpoint names `/fs/usb%0`. The `%0` in the name causes the MCD to try allocate `/fs/usb0`, `/fs/usb1`, `/fs/usb2` and so on (starting form 0) until it finds a unique mountpoint name.

Copyright © 2014, QNX Software Systems Limited

For example, if there are filesystems already mounted at `/fs/usb0` and `/fs/usb1`, then the MCD expands `/fs/usb%0` to `/fs/usb2`.

Below is a sample `mcd.mnt` file that uses the `%#` and `%0` special sequences.

```
#-------------------------------------------------------
# Device                    Mountpt     Type  Options
#-------------------------------------------------------
/dev/cd*                    /fs/cd%#    udf    normv
/dev/umass[0-9]*            /           enum
/dev/umass[0-9]*t1[1234]    /fs/usb%0   dos
/dev/umass[0-9]*t1[1234].*  /fs/usb%0   dos
/dev/umass[0-9]*t[146]      /fs/usb%0   dos
/dev/umass[0-9]*t[146].*    /fs/usb%0   dos
/dev/umass*t7[789]          /fs/usb%0   qnx4
/dev/umass*t17[789]         /fs/usb%0   qnx6  sync=optional
/dev/umass[0-9]*            /fs/usb%0   dos
```

Client API

The MCD uses special rule entries created in the *resource manager filesystem* (p. 1179) to notify client applications of media content matches.

A client application can call *open()* to access the rule entry in which it is interested, and when that rule is matched, it can then use *read()* to read from the entry the name of the relevant mediastore. If QNET is active and the -L is not specified, the device name returned from reading the MCD rule entry is a fully-qualified-path-name (FQPN); this feature allows the string to be used as-is by any process on any node.

The *read()* function blocks until a match is made (unless *oflag* is set to `O_NONBLOCK`). For non-blocking notifications, use *ionotify()*. To wait on multiple rules, use *select()*.

Maintained information

In order to inform each client once and only once of each match, the MCD server maintains state information about each mediastore, matched rule, and client application.

When a new mediastore is inserted, any matched rule triggers notifications to the interested clients. If the mediastore was inserted before a client registered with the MCD, the first read the client makes is satisfied immediately. This behavior eliminates any start-up race conditions, such as, for example, there being media already in a drive at system startup, and the content detection process completing before the higher-level client applications are even started.

Example: Media player

A very simple media player might be designed as follows:

```
int     fd, cd;
char    device[_POSIX_PATH_MAX];

// Open the CD_AUDIO rule and wait for it to be matched.
fd = open("/dev/mcd/CD_AUDIO", O_RDONLY);
while (read(fd, device, sizeof(device)) != -1) {
    // At this point, device contains an audio CD ...
    cd = open(device, O_RDONLY);
    // ... read the toc, play it, etc.
    // Could monitor playback status with
    // DCMD_CAM_CDROMSUBCHNL.
    // If disk is ejected, this will fail.
```

Copyright © 2014, QNX Software Systems Limited

```
    // Can loop back waiting for next insertion.
    // The rule will be re-armed for the
    // next match.
    close(cd);
}
close(fd);
```

Example: Polling

The `mcd` device `Start Rule=` and `Stop Rule=` rule chains are mutually exclusive: the ejection of a device cancels out inserted rules for that device (and vice-versa). Therefore, if you use *select()* or *ionotify()*, you should use them in conjunction with a non-blocking *read()*, as there is no guarantee that the notified state and/or rules of the trigger will remain valid (for example, if the media is ejected between the calls to *ionotify()* and *read()*).

The code snippet below illustrates one way to use *ionotify()* in conjunction with a non-blocking *read()*:

```
<PRE>
   fd = open(rulename, O_RDONLY | O_NONBLOCK);
   SIGEV_UNBLOCK_INIT(&evt);
   for (;;) {
       while (ionotify(fd, _NOTIFY_ACTION_POLLARM, _NOTIFY_COND_INPUT,
                       &evt) != 0) {
           while (read(fd, device, sizeof(device)) > 0) {
               // 'device' matched on 'rulename'
           }
       }
       pause();
   }
</PRE>
```

In a real application, the event would likely be a pulse, and there would be an event-driven loop rather than a pause as in the code snippet above.

Stale Rules

Stale rules may occur if there is client decoupling, and/or a delay between the notification and the use of inserted mediastore; for example, due to the spawning of a separate media application.

To avoid stale rules, the MCD can include the mediastore's insertion sequence number with the rule notification, and applications can then match this number against the device entry in the `/dev/mcd/.devices` directory. If the device has been ejected since the rule was triggered, these values will no longer match, indicating that the rule no longer applies to the current device content, and that new rules may have been re-triggered.

If the client application requires an insertion sequence number, the MCD uses an `XTYPE` read to return an additional `uint32_t` of data with the mediastore name, and the `_IO_XTYPE_MQUEUE` message priority code, avoiding the need to make changes to the global `<sys/io_msg.h>` header file.

```
// Get rule notification using an XTYPE read
int        fd;
uint32_t   seq1;
char       device[_POSIX_PATH_MAX];
```


Copyright © 2014, QNX Software Systems Limited

```
fd = open("/dev/mcd/CD_AUDIO", O_RDONLY);
_readx(fd, device, sizeof(device), _IO_XFLAG_BLOCK | _IO_XTYPE_MQUEUE,
    &seq1, sizeof(seq1));

// Open and check the current version of the inserted device
int         fd;
struct stat st;
uint32_t    seq2;
char        entry[_POSIX_PATH_MAX];

fd = open(device, O_RDONLY);

sprintf(device, "/dev/mcd/.devices/%s", device);
seq2 = (stat(device, &st) != -1) ? st.st_ino : 0;

// If these match, the CD_AUDIO rule is the same and still valid
// and 'fd' is open on that version of the media
if (seq1 == seq2) ...
```

Additional Information

This section describes how to use the MCD for specific operations:

- *Detecting other kinds of system media* (p. 1189)
- *Detecting USB and iPod devices* (p. 1189)
- *Pattern matching and case-sensitivity* (p. 1190)
- *Matching a single rule* (p. 1191)
- *Detecting CD insertion with non-media content* (p. 1191)
- *CD-Changer controlled by external firmware* (p. 1192)
- *Using the MCD as a partition enumerator* (p. 1192)

Detecting other kinds of system media

To detect system media not handled by the routines included with the MCD:

1. Determine what distinguishes your new media type from all other media types. In many cases, the difference might simply be the presence of certain files or directories on the mediastore. For example, a navigation update disk might be identified by the presence of `acios_db.ini` or `config.nfm` files. In this case, a built-in routine such as `FNAME_MATCH`, could perform the detection; or you might have to write a custom routine and provide it to the MCD in an external DLL.
2. Determine the precedence to probe for this media amongst the existing media tests (first, last, after checking for audio content but before checking any other data rules, etc).
3. In the configuration file, make a new rule section for this test, with the appropriate `Callout=` rule, and splice it into the appropriate point of the decision tree by editing the `Match Rule=` or `Fail Rule=` parameters of both the preceding rule and the new rule. The name of the new rule can be used to trigger any client application when the new content is matched.

Detecting USB and iPod devices

The MCD can manage any kind of device, provided that a notification mechanism is available to report on insertions and start the detection process.

Copyright © 2014, QNX Software Systems Limited

For USB devices, you can use the following entry in the MCD configuration file:

```
[/fs/usb*]
Callout     = PATH_MEDIA_PROCMGR
Argument    = /proc/mount
Priority    = 11,10
Start Rule  = ...
```

Targets running QNX Neutrino 6.3.*n* releases only. For USB devices, the `umass-enum` server in conjunction with the MCD's built-in `USB_MEDIA_ENUM` routine can provide the notification mechanism and start the detection process. Invoke `umass-enum` with the -r option to activate its resource manager interface, and use the following device entry in the MCD configuration file:

```
[/dev/umass/*]
Callout     =   USB_MEDIA_ENUM
Argument    =   /dev/umass-enum
Priority    =   11,10
Start Rule  =   ...
```

For iPod devices, the device entries dynamically attach pathnames (when these pathnames are present), and so can be handled with the MCD's built-in *PATH_MEDIA_PROCMGR()* routine. Use the following device entry in the MCD configuration file for iPods:

```
[/fs/ipod*]
Callout     =   PATH_MEDIA_PROCMGR
Argument    =   /proc/mount
Priority    =   11,10
Start Rule  =   ...
```

Pattern matching and case-sensitivity

The MCD's `FNAME_MATCH` routine attempts to access listed files by using the underlying filesystem, which applies any rules appropriate for that filesystem *in conjunction with* any specified mount options. Thus, case-sensitivity in pattern matching depends on:

- the built-in routine used
- the underlying filesystem

The table below lists the case-sensitivity and case-preserving characteristics of some common filesystems:

filesystem	case-sensitive	case-preserving
FAT	No	No
ISO-9660	No	No
Joliet	No	Yes
QNX 4	Yes	Yes
RRIP	Yes	Yes

Copyright © 2014, QNX Software Systems Limited

filesystem	case-sensitive	case-preserving
VFAT	No	Yes

Since the majority of mediastores used for multimedia storage are formatted for DOS/Windows use, it is likely that the filesystem will be case-insensitive. This case-insensitivity means that in any FNAME_MATCH rules the Argument= filename list can be given in either upper or lower case.

The FNAME_PATTERN routine processes directory entries from the filesystem through the libc *fnmatch()* function, which is case sensitive.

The directory output from each filesystem depends on whether it is case-preserving. If the filesystem is *not* case-preserving, then default rules are used to control the filename presentation. Refer to the cd case=lower|upper or the dos sfn=lower|upper|windows filesystem mount options.

Since the most common multimedia formats are case-preserving and will use the exact filename that a user or media application used to create their files, in any FNAME_PATTERN rules the Argument= pattern list should be given in both upper and lower case (as in the MIXED_AV rule in the *Examples* (p. 1193) below.).

Matching a single rule

If you don't want multiple media types to match and only want to match the first rule, you can use the fact that if a matched rule has no Match Rule= branch the MCD stops its detection process.

In the configuration file, simply arrange the rules, from the Start Rule= through the Fail Rule= links, in priority order. Do not provide any Match Rule= branches. The MCD detection framework will test these rules in sequence until one is matched, then stop. For an example, see the VIDEO_CD and SVIDEO_CD entries in the *Examples* (p. 1193) below.

Detecting CD insertion with non-media content

To be notified when a CD is inserted regardless of what content it contains, simply:

- Make a dummy rule with a no Callout= routine.
- Make a Match Rule= branch (so that it will always match when tested).
- Make this rule the Start Rule= of the device (with your dummy rule branching to the original start rule).

Your application can now block on that new rule, via the normal resource manager interface ("/dev/mcd/DISC_INSERTED"), waiting for device insertion. For example, one of the example configurations in this document could be modified as follows:

```
[/dev/cd0]
Callout      =    CD_MEDIA_IOBLK
Argument     =    1000,2000
Priority     =    11,9
```

Copyright © 2014, QNX Software Systems Limited

```
Start Rule   =    DISC_INSERTED

[DISC_INSERTED]
Match Rule   =    DVD_OR_CD
```

CD-changer controlled by external firmware

To detect insertion events from a CD-changer that is controlled by external firmware (e.g. FJ-10), you should not use any of the built-in MCD detection callouts, but trigger the insertion notification directly from the CD-changer controller stack.

Internally, all built-in device detection callouts do their specific work, then write the name of the device to the special `/dev/mcd/.insert` entry. This behavior means that to detect insertions on changers controlled by external firmware, proceed as follows:

- In the device driver, wait for the CD-changer to be loaded and available for use by the system
- When the CD-changer is available, write the name of the device (as a string, i.e. `/dev/cd0`) to the MCD control point. Writing the name of the device to the control point triggers the rest of the content detection process.

In the configuration file, the relevant device entry would be like this (note that no `Callout=` is specified in this situation):

```
[/dev/cd0]
Priority     =    11
Start Rule   =    ...
```

The insertion notification code in the driver is basically this:

```
int    notify;

notify = open("/dev/mcd/.insert", O_WRONLY);
write(notify, "/dev/cd0", 8);
close(notify);
```

Similar code to handle ejections can be written to `/dev/mcd/.eject`.

Using the MCD as a partition enumerator

The MCD's `MOUNT_FSYS` rule can be used to determine if partition enumeration has occurred on a device (USB stick). This capability can be used to try to mount a filesystem on the raw device, if the device has not been partitioned.

In order to determine if partition has occurred, use the following configuration:

- Configure the USB enumerator to start `devb-umass` with the blk auto=none option; that is, to *not* automatically enumerate partitions.
- Configure the MCD device rule for the USB as follows:
  - it must be of the form `[/dev/umass*]`; that is, so that the pattern matches both the raw device and any partitions
  - its `Start Rule=` should be a rule that invokes `MOUNT_FSYS`

Copyright © 2014, QNX Software Systems Limited

- Configure a second-phase set of mountpoint rules `[/fs/usb*]` to continue processing once a filesystem has been mounted from either a partition or a device.

  The `mcd.mnt` rule used by that `MOUNT_FSYS` should include the following:

```
/dev/umass[0-9]*         /           enum
/dev/umass[0-9]*         /fs/usb%#   dos
/dev/umass[0-9]*t1[124] /fs/usb%#   dos     fsi=use
/dev/umass[0-9]*t[146]  /fs/usb%#   dos
```

The control flow for this configuration is as follows:

1. USB stick inserted.
2. USB enumerator detects insertion and launches `devb-umass`.
3. `devb-umass` puts up `/dev/umass`*X* pathname, triggering the MCD.
4. The MCD runs the `MOUNT_FSYS` rule.
5. If the media is non-partitioned:
   a. The `enum` rule is executed and fails.
   b. The code falls through and attempts `fs-dos` mount on raw device, which should succeed, resulting in the appearance of an `/fs/usb*`.
6. If the media is partitioned:
   a. The `enum` rule enumerates partitions and, thus, succeed, terminating the callout.
   b. The enumeration makes `/dev/umass`*X*`t`*N* names appear, re-entering the MCD device rule with a pattern that skips the `enum` rules, and instead tries `fs-dos` mounts on a partition, resulting in appearance of an `/fs/usb*`.
7. Following either of the above two cases (5 or 6), `MOUNT_FSYS` is successful with a mount, and the MCD continues with `/fs/usb*` rules, typically some form of content detection or the triggering of a dummy `INSERT` rule.

X is the disk number (0, 1, 2, etc.). and *N* is the partition type (4, 11, 12, etc.); for example: `/dev/umass[0-9]*` or `/dev/umass[0-9]*t1[124]`. Thus, a path with `umass`*X* refers to a device, while a path with `umass`*X*`t`*N* refers to a partition.

Examples:

Consider the following sample configuration file for a CD-based system:

```
# Sample CD/DVD disk identification rules.

[/dev/cd0]
Callout     =   CD_MEDIA_IOBLK
Argument    =   1000,2000
Priority    =   11,9
Start Rule  =   DVD_OR_CD

[DVD_OR_CD]
Callout     =   DVD_OR_CD
Match Rule  =   DVD_AUDIO
Fail Rule   =   CD_AUDIO

[DVD_AUDIO]
```

Copyright © 2014, QNX Software Systems Limited

```
Callout     =   FNAME_MATCH
Argument    =   /AUDIO_TS/AUDIO_TS.IFO
Match Rule  =   DVD_VIDEO
Fail Rule   =   DVD_VIDEO

[DVD_VIDEO]
Callout     =   FNAME_MATCH
Argument    =   /VIDEO_TS/VIDEO_TS.IFO
Fail Rule   =   VIDEO_CD

[CD_AUDIO]
Callout     =   CD_AUDIO
Match Rule  =   VIDEO_CD
Fail Rule   =   VIDEO_CD

[VIDEO_CD]
Callout     =   FNAME_MATCH
Argument    =   /VCD/INFO.VCD,/MPEGAV/AVSEQ01.DAT,/MPEGAV/MUSIC01.DAT
Fail Rule   =   SVIDEO_CD

[SVIDEO_CD]
Callout     =   FNAME_MATCH
Argument    =   /SVCD/INFO.SVD,/MPEGAV/AVSEQ01.MPG,/MPEG2/AVSEQ01.MPG
Fail Rule   =   MIXED_AV

[MIXED_AV]
Callout     =   FNAME_PATTERN
Argument    =   *.MP3,*.mp3,*.WMV,*.wmv,*.WMA,*.wma,*.AAC,*.aac,*.JPG,*.jpg,*.MPG,*.mpg
```

A single device, `/dev/cd0`, is monitored by the built-in *CD_MEDIA_IOBLK()* routine:

- When media is inserted, the content detection process begins with the `DVD_OR_CD` rule.
- If this rule matches (is a DVD) then the process branches to the `DVD_AUDIO` rule. If this rule fails, the process branches to the `CD_AUDIO` rule.

  The `DVD_AUDIO` rule assumes the existence (using the built-in `FNAME_MATCH` test) of a `/AUDIO_TS/AUDIO_TS.IFO` file on the DVD indicating DVD audio content.
- Since both audio and video content may be present on a DVD, both Match and Fail branch from here to the same rule to try next: `DVD_VIDEO`.

  This behavior is similar to the behavior of the `CD_AUDIO` rule, as a CD can contain both audio and data content.

  Rules for determining CD data content, such `VIDEO_CD` or `SVIDEO_CD`, have only a Fail branch, since a match at these levels is mutually exclusive with any further content. If these rules match, then the content detection process stops.
- The final `MIXED_AV` rule has no branches. Regardless of the outcome, processing stops at this rule.

During the content detection processing phase, any clients that have registered against any matched rules will be notified. Multiple rules, or no rules at all, might be matched by an inserted CD: an enhanced audio CD with a bonus music video might match both `CD_AUDIO` and `MIXED_AV` rules, whereas a data CD backup of a development system would match none.

Two-phase filesystem mount example

Below is a configuration file involving USB devices; it requires that an external USB enumerator invoke `devb-umass blk auto=partition disk name=umass` in response to insertions.

```
[/dev/umass*t*]
Callout     =   PATH_MEDIA_PROCMGR
Argument    =   /proc/mount
```


Copyright © 2014, QNX Software Systems Limited

```
Priority    =   11,10
Start Rule  =   MOUNT

[MOUNT]
Callout     =   MOUNT_FSYS
Argument    =   /dev/shmem/mcd.mnt

[/fs/usb*]
Callout     =   PATH_MEDIA_PROCMGR
Argument    =   /proc/mount
Priority    =   11,10
Start Rule  =   MIXED_AV

[MIXED_AV]
Callout     =   FNAME_PATTERN
Argument    =   *.MP3,*.mp3,*.WMV,*.wmv,*.WMA,*.wma,*.AAC,*.aac,*.JPG,*.jpg,*.MPG,*.mpg
```

Note that the device pattern specified in the example above avoids the raw device itself and only applies to partition entries. The mount configuration `/dev/shmem/mcd.mnt` referred to contains:

```
/dev/umass*t1[124]      /fs/usb%#   dos
/dev/umass*t[146]       /fs/usb%#   dos
```

Copyright © 2014, QNX Software Systems Limited

mcs

Manipulate the comment section of an object file

Syntax:

```
mcs [-cdpV] [-a string] [-n name]* file...
```

Runs on:

QNX Neutrino

Options:

-a *string*

Append *string* to the comment section.

-c

Compress the contents of the comment section.

-d

Delete the comment section.

-n *name*

Specify the name of the comment section. The default is `.comment`.

-p

Print the contents of the comment section.

-V

Print the program version.

Description:

The `mcs` utility manipulates the comment section of an object file.

Copyright © 2014, QNX Software Systems Limited

melt

Uncompress frozen files (UNIX)

Syntax:

```
melt [-cfvVz] [filename...]
```

Runs on:

QNX Neutrino

Options:

See *freeze* (p. 784).

Description:

The `melt` utility is equivalent to `freeze -d`. See the `freeze` utility for details.

The `freeze`/`melt`/*fcat* (p. 729) compression utilities will eventually become deprecated in favor of the GNU `zip` suite, *gzip* (p. 921)/*gunzip* (p. 920)/*zcat* (p. 2080). The `freeze` suite of utilities will continue to be provided for quite some time before being eliminated completely.

Contributing author:

Leonid A. Broukhis

Copyright © 2014, QNX Software Systems Limited

mesg

Enable, disable, or query broadcast messages (QNX Neutrino)

Syntax:

```
mesg [toggle]
```

Runs on:

QNX Neutrino

Options:

Where *toggle* is one of:

n

Disable broadcast messages on the console.

y

Enable broadcast messages on the console.

The exit values are:

- 0 - Receiving messages is allowed.
- 1 - Receiving messages is not allowed.
- > 1 - An error occurred.

If no *toggle* is specified, `mesg` prints the current status.

Description:

The `mesg` utility enables, disables, or queries broadcast messages on the console.

Copyright © 2014, QNX Software Systems Limited

mkasmoff

Create an assembler include file from an ELF or COFF file

Syntax:

```
mkasmoff infile outfile
```

Runs on:

Linux, Microsoft Windows

Options:

infile

The name of an input ELF or COFF file.

outfile

The name of the output file.

Description:

The `mkasmoff` utility reads the *infile* ELF or COFF object file (created with the macros in `<mkasmoff.h>`) and creates an assembler include file called *outfile* containing the appropriate macro definitions.

Copyright © 2014, QNX Software Systems Limited

mkcldr

Convert CLDR text file to binary data for `libqdb_cldr.so`

Syntax:

```
mkcldr [options]* input file output file
```

Runs on:

QNX Neutrino

Targets:

ARM, x86

Options:

-c

Name of the CHARMAP file (typically `posix/UTF-8.cm` from CLDR).

-i

Secondary weight/flag for case inversion (default 0x0080).

-n

Override name of collation in output file header.

-w

Set variable weight threshold for ignorables (punctuation).

Description:

The `mkcldr` utility converts standard CLDR language collation (or sort order) tables into a format usable by the `libqdb_cldr.so` DLL. Specifically, it reads the contents of a text input file in the CLDR (Common Locale Data Repository) POSIX format and writes this content to an output file as binary data suitable for `libqdb_cldr.so`.

To use other language collation tables, you can download them from *`cldr.unicode.org`*, then use `mkcldr` to convert them.

If the collation rules in the files shipped with the OS or in the downloaded files don't correspond to the rules required for your locale or implementation, you can adjust them, then use `mkcldr` to create the files with the binary data format required by `libqdb_cldr.so`.

Copyright © 2014, QNX Software Systems Limited

Examples:

The following example converts the file for German used in Switzerland:

```
$ cd cldr-1.4.1/posix
$ mkcldr -c UTF-8.cm de_CH.UTF-8.src /etc/cldr/de_CH
```

The `UTF-8.cm` file is simply a database that maps textual character descriptions to their Unicode value; it is used in parsing the collation information.

Copyright © 2014, QNX Software Systems Limited

mkdir

Make directories (POSIX)

Syntax:

```
mkdir [-m mode] [-p] dir...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-m *mode*

When creating the directory, set the permission bits of the new directory to the specified *mode* value.

The *mode* argument is a *symbolic_mode* string, as defined for the *chmod* (p. 124) utility. In the *symbolic_mode* strings, the *op* characters + and - are interpreted relative to the default file mode for that file type:

+

Add permissions to the default mode.

-

Delete permissions from the default mode.

=

Assign permissions.

-p

Create any missing intermediate pathname components.

dir

A pathname at which a directory is to be created.

If you specify both the -p and -m options, any intermediate directories you have created have mode u+wx.

Copyright © 2014, QNX Software Systems Limited

Description:

The `mkdir` utility creates the directories specified by the *dir* operands, in the order the *dir* operands are specified.

To create a directory, you must have write permission on the parent directory, or be `root`.

Not all filesystems support the creation of directories. For example, `/dev/shmem` (which really isn't a filesystem but looks like one) doesn't. For more information, see the Working with Filesystems chapter of the QNX Neutrino *User's Guide*.

The default file mode for directories is `a=rwx` (777), with selected permissions removed in accordance with the file mode creation mask (see the *umask* (p. 2005) utility).

For intermediate pathname components created by `mkdir`, the mode is the default modified by `u+wx` so that the subdirectories can always be created regardless of the file-mode creation mask. If you want to assign different ultimate permissions for the intermediate directories, you can do so with the *chmod* (p. 124) utility.

When using -p with -m, each intermediate directory that doesn't exist is created with `u+wx` modes, regardless of the file mode creation mask. The specified *mode* applies only to the last directory specified. For example:

```
mkdir -p -m 777 dir/dir1/dir2
```

gives `dir` and `dir1` the default permissions for intermediate directories (i.e. `u+wx`). The directory `dir2` is given `a+rwx` permission.

The default file-creation mask influences the behavior of `mkdir`.

Examples:

Create a directory named `/home/debbie`:

```
mkdir /home/debbie
```

Exit status:

0

All the specified directories were created successfully, or the -p option was specified and all the specified directories now exist.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Caveats:

If the `mkdir` utility is terminated by a signal, some of the specified directories may have already been created.

Copyright © 2014, QNX Software Systems Limited

mkdosfs

Format a DOS (FAT-12/16/32) filesystem (QNX Neutrino)

Syntax:

```
mkdosfs [-C|c size] [-e number]
        [-F type] [-f number] [-h number]
        [-I vol_id] [-L vol_label] [-m media]
        [-O oem_label] [-R|r] [-S size] [-s size]
        device | mountpoint | file
```

Runs on:

QNX Neutrino

Options:

-C *size*

Minimum default cluster size. Don't specify both -C and -c.

-c *size*

Cluster size for the filesystem; the default is determined by disk geometry. Don't specify both -c and -C.

-e *number*

Number of root directory entries (FAT12/16 only); the default is 512.

-F *type*

The FAT type (12, 16, or 32).

-f *number*

Set the number of FAT tables to write; the default is 2.

-h *number*

The number of “hidden sectors”; the default is determined by disk geometry.

-I *vol_id*

Set the volume ID/serial number; the default is based on the current time.

-L *vol_label*

Specify a volume label; the default is none.

Copyright © 2014, QNX Software Systems Limited

-m ***media***

Media descriptor byte; the default is `0xF0` or, if there are hidden sectors, `0xF8`.

-O ***oem_label***

Set the OEM label; the default is:

MSDOS5.0

For a FAT12/FAT16 filesystem.

MSWIN4.1

For a FAT32 filesystem.

-R

Preserve the size and content of existing reserved sectors (reformat).

-r

The number of "reserved sectors"; the default is determined by FAT type.

-S ***size***

Sector size for the filesystem; the default is determined by disk geometry.

-s ***size***

Set the size (number of sectors) in the filesystem; the default is determined by disk geometry.

device

The device name to host the DOS filesystem (e.g. `/dev/hd0t11`).

mountpoint

The mountpoint of the DOS filesystem (e.g. `/fs/hd0-dos`).

file

A regular file to contain the DOS filesystem image.

Description:

The `mkdosfs` utility formats a DOS filesystem on the specified target (typically a disk device or partition).

If you don't specify essential user options such as FAT type and cluster size, `mkdosfs` formats the DOS filesystem using the most suitable options for the size and disk

Copyright © 2014, QNX Software Systems Limited

geometry of the host. You can override this default auto-configuration and force a particular format to be used by setting the options you need.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

```
mkdosfs /dev/hd0t6
Format complete: FAT16 (4096-byte clusters), 201888 kB available.
```

Exit status:

0

The filesystem was constructed without error.

Copyright © 2014, QNX Software Systems Limited

1

The filesystem wasn't constructed. This may be due to an error or inconsistency with the user options or because of a nonrecoverable error, such as disk I/O or insufficient memory.

Contributing author:

Robert Nordier

License:

This utility is based on software from Robert Nordier; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Caveats:

The `mkdosfs` utility destroys or overwrites any existing filesystem on the target.

Copyright © 2014, QNX Software Systems Limited

mkefs

Build an embedded filesystem (QNX)

Syntax:

```
mkefs [-c cache_dir] -t ffs2 | ffs3 [-l inputline] [-nv]
      [buildfile [outputfile]]
```

Runs on:

Linux, Microsoft Windows

Options:

-c *cache_dir*

Cache compressed files in *cache_dir*.

-l *inputline*

("el") Process *inputline* before interpretation of the *buildfile* begins. Input lines given to `mkefs` should be quoted to prevent interpretation by the shell (as `mkefs` input lines often contain spaces). Multiple -l options are processed in the order specified. No default.

-n

Don't use timestamps in the files. Using the -n option permits identical images in binary format. Specifying additional -n options strips all time information from files.

-t ffs2 | -t ffs3

Set the type of output file system. Use -t ffs2 to make a version 2 flash filesystem image. Use -t ffs3 to make a version 3 flash filesystem image. The default is ffs3.

-v

Operate verbosely. Specifying additional -v options increases verbosity. Default is quiet operation.

Description:

The `mkefs` utility reads a text buildfile describing an embedded filesystem and produces a binary image file containing the embedded filesystem. You can copy this

Copyright © 2014, QNX Software Systems Limited

file to flash at a later stage, or use *mkimage* (p. 1273) to combine it with an OS image before downloading.

Don't confuse this command with *mkifs* (p. 1241), which builds an OS *image* filesystem, or *mketfs* (p. 1219), which builds an embedded transaction filesystem (ETFS).

The input and output files are specified on the command line:

buildfile

The filename of the buildfile that describes the contents of the embedded filesystem; use "-" to specify standard input (the default).

outputfile

The filename of the image file containing the embedded filesystem; use "-" to specify standard output (the default). Note that you can specify the *outputfile* only if you specified a *buildfile*.

QNX Neutrino flash filesystem version 3 no longer provides built-in decompression. The flash filesystem's decompression functionality has moved into the *inflator* (p. 984) resource manager. You should now use the *deflate* (p. 183) utility to compress files.

Buildfiles

The buildfile uses the same grammar as the *mkifs* (p. 1241) command, but supports different attributes.

The buildfile is basically just a list of files that you want to be included in the embedded filesystem image file when it's built by `mkefs`. As well as the files to be included, you can specify various attributes that are used to set parameters of the filesystem and the files in it. For example, you can specify the maximum size of the filesystem, or the user and group IDs of the individual files.

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```


Copyright © 2014, QNX Software Systems Limited

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

```
[attribute] A
B
C
```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

```
[attribute]
A
B
C
```

Attributes provide information about the file following the attribute. They're enclosed in square brackets; when combining attributes (e.g. to specify both the user ID *and* the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
[uid=5 gid=5] filename

# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

boolean attributes

Those prefixed with a plus ("+") or minus ("-") sign.

value attributes

Those ending with an equals sign ("=") followed by a value. Don't put any spaces around the equals sign.

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example, `?+bigendian` sets the `+bigendian` attribute only if `+bigendian` or `-bigendian` hasn't already been set.

The *file_specification* takes one of the following forms:

path

Copyright © 2014, QNX Software Systems Limited

The file is copied from the host to the location in the image defined by the *prefix* (p. 1217) attribute. If *path* isn't absolute, `mkefs` looks for it in the locations identified by the *search* (p. 1217) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces ({ }) — the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mkefs` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mkefs` doesn't interpret a pound sign (#) in an inline file as the beginning of a comment. The syntax of the inline file depends on what it's used for on the target system.

Closing braces (}) and backslashes (\) in an inline file must be escaped with a backslash.

You can enclose a filename in double quotes ("") if it includes spaces or unusual characters.

Attributes

The `mkefs` command supports the following attributes:

- *+/-bigendian* (p. 1213)
- *block_size=bsize* (p. 1213)
- *cd=filename* (p. 1213)
- *dperms=dperm_spec* (p. 1213)
- *ecc_on=value* (p. 1214)
- *filter=filter_spec* (p. 1214)
- *+/-followlink* (p. 1214)
- *gid=id_spec* (p. 1215)
- *max_size=msize* (p. 1215)
- *min_size=tsize* (p. 1215)
- *mount=filename* (p. 1215)
- *mountperms=perm_spec* (p. 1215)
- *mtime=time_spec* (p. 1216)

Copyright © 2014, QNX Software Systems Limited

- *+|-optional* (p. 1216)
- *perms=perm_spec* (p. 1216)
- *prefix=prefix_spec* (p. 1217)
- *search=path:path...* (p. 1217)
- *spare_blocks=sblocks* (p. 1217)
- *type=file_type* (p. 1217)
- *uid=id_spec* (p. 1218)

An OR-bar indicates that either the first element or the second element must be present, but not both (e.g. +|- `bigendian` means either `+bigendian` or `-bigendian`, but not `+-bigendian`).

`bigendian` attribute (boolean)

`+|-bigendian`

Set the byte order for the embedded filesystem to either big (via `+bigendian`) or little (via `-bigendian`) endian. The default is little endian.

`block_size` attribute

`block_size=`*`bsize`*

Set the block size for the embedded filesystem. The block size depends on what memory devices you have in your target hardware and how they're arranged. For example, two interleaved 64K × 8-bit devices configured for a 16-bit interface have a block size of 128K bytes. The default block size is 64K bytes.

`cd` attribute

`cd=`*`filename`*

Set the current working directory to the specified pathname before attempting to open the host file. Default is the directory from which `mkefs` was invoked.

`dperms` attribute

`dperms=`*`dperms_spec`*

Set the access permissions of the directory. If specified as a number, the permissions are set to that number (just like the `chmod` command). If specified as an asterisk ("*"), the host directory's permissions are used; for an inline directory, the permissions are obtained from the `umask` of the user running `mkefs`. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`

Copyright © 2014, QNX Software Systems Limited

3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *dperms_spec* is "*".

ecc_on attribute

`ecc_on=`*`value`*

Control error-correcting code (ECC) support:

Specify *value* as:	To:
0	Disable ECC (the default)
1	Enable 64-byte alignment ECC
2	Enable 32-byte alignment ECC

Don't mix ECC-enabled partitions and ECC-disabled partitions; the `devf-*` drivers don't support this.

The `mkefs` utility automatically handles the formatting alignment, but you have to use the -b option for *`flashctl`* (p. 771) to specify the appropriate alignment.

`filter` attribute

`filter=`*`filter_spec`*

Run the host file through the filter program specified, presenting the host file data as standard input to the program and using the standard output from the program as the data to be placed into the embedded filesystem. Default is no filter.

The most common filter you're likely to use with the embedded filesystem is *`deflate`* (p. 183). This compresses the file before copying it to the embedded filesystem. For example:

```
[filter="deflate"]
```

You can specify a *filter_spec* of `none`. This is useful if you need to override a global filter specification.

`followlink` attribute (boolean)

`[+|-followlink]target_path=host_path`

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mkefs` resolves the symbolic link and includes its target file or directory. If you specify `-followlink`, `mkefs` includes the symbolic

Copyright © 2014, QNX Software Systems Limited

link itself in the embedded filesystem. It's up to you to include in the image whatever the link points to.

gid attribute

gid=*id_spec*

Set the group ID number for the file. The value of this attribute may be either a number or an asterisk (*). If it's an asterisk, the group ID is taken from the host file; for an inline file, the group ID is the group of the user running mkefs. The default value for this attribute is *.

max_size attribute

max_size=*size*

Set the maximum size of the embedded filesystem. You can set this attribute if you don't want the filesystem to exceed a maximum size. If this occurs while mkefs is building the filesystem, you'll see a warning message. The default is 4G bytes.

min_size attribute

min_size=*size*

Set the minimum size of the embedded filesystem. If the size of the filesystem is less than this size after all the specified files have been added, then the filesystem is padded to the required size. The default is unspecified, meaning that no padding occurs.

mount attribute

mount=*filename*

Specify the mountpoint for the embedded filesystem. You can override the mountpoint with the -n option to the *flashctl* (p. 771) command.

The default is "", which makes the flash filesystem drivers (*devf-** (p. 169)) use the appropriate default, usually /fs1p1.

mountperms attribute

mountperms=*perms_spec*

Set the access permissions for mountpoints. If specified as a number, the permissions are set to that number (just like the chmod command). Otherwise, a symbolic mode string (which is a subset of chmod's) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of u, g, o, and a
2. one of -, =, or +
3. a combination of r, w, x, s, g, and t.

Copyright © 2014, QNX Software Systems Limited

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *perms_spec* is "0777".

`mtime` attribute

```
mtime=time_spec
```

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (`*`), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```

  You must provide all six elements. The time is always interpreted as UTC.

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

`optional` attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the embedded filesystem. If false, and the host file can't be found, output an error message and exit `mkefs`. The default is true.

`perms` attribute

```
perms=perms_spec
```

Set the access permissions of the file. If specified as a number, the permissions are set to that number (just like the `chmod` command). If specified as an asterisk ("*"), the host file's permissions are used; for an inline file, permissions of 0666 are used. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *perms_spec* is "*".

Copyright © 2014, QNX Software Systems Limited

When running on a Windows host, `mkefs` can't get the execute (`x`), setuid ("set user ID"), or setgid ("set group ID") permissions from the file. Use the `perms` attribute to specify these permissions explicitly. You might also have to use the *`uid`* (p. 1218) and *`gid`* (p. 1215) attributes to set the ownership correctly. To determine whether or not a utility needs to have the setuid or setgid permission set, see its entry in the *Utilities Reference*.

`prefix` attribute

`prefix=`*`prefix_spec`*

Set the prefix on the target file names. The default is the empty string.

`search` attribute

`search=`*`path`*`:`*`path`*`:…`

This attribute specifies that `mkefs` should search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the image file system.

Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

`spare_blocks` attribute

`spare_blocks=`*`sblocks`*

Set the number of spare blocks to be used by the embedded filesystem. If you want the embedded filesystem to be able to reclaim the space taken up by deleted files, set the number of spare blocks to 1 or more. The default is 1.

`type` attribute

`type=`*`file_type`*

Sets the type of the files being created in the embedded filesystem. Allowable types are:

- `link` — a symbolic link
- `file` — a regular, everyday file (the default)
- `dir` — a directory.

Specifying `[type=dir]` tells `mkefs` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

Copyright © 2014, QNX Software Systems Limited

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```

`uid` attribute

`uid=`*`id_spec`*

Set the user ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the user ID is taken from the host file; for an inline file, the user ID is the user running `mkefs`. The default value for this attribute is "`*`".

Examples:

Here's a sample buildfile, `my_efs.bld`:

```
# A sample buildfile for mkefs

[block_size=128k spare_blocks=1]
/home/jwall/nto_flash
```

In this example, we've specified a block size of 128K and one spare block. The files and subdirectories from the `/home/jwall/nto_flash` directory on the host system are to be recursively copied into the root directory of the embedded filesystem.

To create an embedded filesystem image file using the above buildfile, invoke `mkefs` as follows:

```
mkefs my_efs.bld my_image.efs
```

This creates the `my_image.efs` file containing the embedded filesystem, which can then be copied to the target system.

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mketfs

Build an embedded transaction filesystem (QNX)

Syntax:

```
mketfs  [-l inputline] [-nv] [buildfile [outputfile]]
```

Runs on:

Linux, Microsoft Windows

Options:

-l inputline

("el") Process *inputline* before interpretation of the *buildfile* begins. Input lines given to `mketfs` should be quoted to prevent interpretation by the shell (as `mketfs` input lines often contain spaces). Multiple -l options are processed in the order specified. No default.

-n

Don't use timestamps in the files. Using the -n option permits identical images in binary format. Specifying additional -n options strips all time information from files.

-v[v..]

Operate verbosely. Specifying additional -v options increases verbosity. Default is quiet operation.

Description:

The `mketfs` utility reads a text buildfile describing an embedded transaction filesystem (ETFS) and produces a binary image file containing the ETFS as a sequence of transactions. You can copy this file to flash at a later stage, using *`etfsctl`* (p. 713).

Don't confuse this command with *`mkifs`* (p. 1241), which builds an OS *image* filesystem, or *`mkefs`* (p. 1209), which builds an embedded filesystem.

The input and output files are specified on the command line:

buildfile

Copyright © 2014, QNX Software Systems Limited

The filename of the buildfile that describes the contents of the embedded filesystem; use "-" to specify standard input (the default).

outputfile

The filename of the image file containing the ETFS; use "-" to specify standard output (the default). Note that you can specify the *outputfile* only if you specified a *buildfile*.

The `mketfs` utility generates a list of clusters, which are sized to match the hardware (1024, 2048, or 4096 bytes). Each cluster has a header of type `struct etfs_trans`, which is a fixed 16 bytes. So the actual size of the image file is the size of the input data (rounded up to a multiple of the cluster size) plus 16 bytes times the number of clusters. When *`etfsctl`* (p. 713) writes to the filesystem, it removes the `etfs_trans` structure. As the `devio` layer is putting the cluster into hardware, it generates a new, BSP-specific structure to hold the same information as was in the `etfs_trans`.

Buildfiles

The buildfile uses the same grammar as the *`mkifs`* (p. 1241) command, but supports different attributes.

The buildfile is basically just a list of files that you want to be included in the ETFS image file when it's built by `mketfs`. As well as the files to be included, you can specify various attributes that are used to set parameters of the filesystem and the files in it. For example, you can specify the maximum size of the filesystem, or the user and group IDs of the individual files.

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

  ```
  [attribute] A
  ```


Copyright © 2014, QNX Software Systems Limited

```
B
C
```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

```
[attribute]
A
B
C
```

Attributes provide information about the file following the attribute. They're enclosed in square brackets; when combining attributes (e.g. to specify both the user ID *and* the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
[uid=5 gid=5] filename

# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

boolean attributes

Those prefixed with a plus ("+") or minus ("-") sign.

value attributes

Those ending with an equals sign ("=") followed by a value. Don't put any spaces around the equals sign.

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example, `?+bigendian` sets the `+bigendian` attribute only if `+bigendian` or `-bigendian` hasn't already been set.

The *file_specification* takes one of the following forms:

path

The file is copied from the host to the location in the image defined by the *prefix* (p. 1226) attribute. If *path* isn't absolute, `mketfs` looks for it in the locations identified by the *search* (p. 1226) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

Copyright © 2014, QNX Software Systems Limited

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces ({ }) — the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mketfs` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mketfs` doesn't interpret a pound sign (#) in an inline file as the beginning of a comment. The syntax of the inline file depends on what it's used for on the target system.

Closing braces (`}`) and backslashes (`\`) in an inline file must be escaped with a backslash.

You can enclose a filename in double quotes (`""`) if it includes spaces or unusual characters.

Attributes

The `mketfs` command supports the following attributes:

- *+/-`bigendian`* (p. 1223)
- *`block_size`=bsize* (p. 1223)
- *`cd`=filename* (p. 1223)
- *`cluster_size`=csize* (p. 1223)
- *`dperms`=dperm_spec* (p. 1223)
- *`filter`=filter_spec* (p. 1224)
- *+/-`followlink`* (p. 1224)
- *`gid`=id_spec* (p. 1224)
- *`mountperms`=perm_spec* (p. 1224)
- *`mtime`=time_spec* (p. 1224)
- *`num_blocks`=num* (p. 1225)
- *+/-`optional`* (p. 1225)
- *`perms`=perm_spec* (p. 1225)
- *`prefix`=prefix_spec* (p. 1226)
- *`search`=path:path...* (p. 1226)
- *`type`=file_type* (p. 1226)
- *`uid`=id_spec* (p. 1227)

You should explicitly specify the `cluster_size`, `block_size` and `num_blocks` attributes as appropriate for your flash device to ensure that the image produced is fully compatible with your specific device.

Copyright © 2014, QNX Software Systems Limited

An OR-bar indicates that either the first element or the second element must be present, but not both (e.g. +|- `bigendian` means either `+bigendian` or `-bigendian`, but not `+-bigendian`).

`bigendian` attribute (boolean)

`+|-bigendian`

Set the byte order for the embedded filesystem to either big (via `+bigendian`) or little (via `-bigendian`) endian. The default is little endian.

`block_size` attribute

`block_size=`*`bsize`*

Set the block size for the ETFS. The block size depends on what memory device you have in your target hardware. The default block size is 16 KB.

`cd` attribute

`cd=`*`filename`*

Set the current working directory to the specified pathname before attempting to open the host file. Default is the directory from which `mketfs` was invoked.

`cluster_size` attribute

`cluster_size=`*`csize`*

Set the cluster size for the ETFS. The cluster size depends on what memory device you have in your target hardware. The default cluster size is 1 KB.

`dperms` attribute

`dperms=`*`dperms_spec`*

Set the access permissions of the directory. If specified as a number, the permissions are set to that number (just like the `chmod` command). If specified as an asterisk (`*`), the host directory's permissions are used; for an inline directory, the permissions are obtained from the `umask` of the user running `mketfs`. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *dperms_spec* is `*`.

Copyright © 2014, QNX Software Systems Limited

`filter` attribute

`filter=`*`filter_spec`*

Run the host file through the filter program specified, presenting the host file data as standard input to the program and using the standard output from the program as the data to be placed into the embedded filesystem. Default is no filter.

You can specify a *filter_spec* of `none`. This is useful if you need to override a global filter specification.

`followlink` attribute (boolean)

`[+|-followlink]`*`target_path=host_path`*

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mketfs` resolves the symbolic link and includes its target file or directory. If you specify `-followlink`, `mketfs` includes the symbolic link itself in the filesystem. It's up to you to include in the image whatever the link points to.

`gid` attribute

`gid=`*`id_spec`*

Set the group ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the group ID is taken from the host file; for an inline file, the group ID is the group of the user running `mketfs`. The default value for this attribute is `*`.

`mountperms` attribute

`mountperms=`*`perms_spec`*

Set the access permissions for mountpoints. If specified as a number, the permissions are set to that number (just like the `chmod` command). Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *perms_spec* is “0775”.

`mtime` attribute

`mtime=`*`time_spec`*

Copyright © 2014, QNX Software Systems Limited

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (*), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```

  You must provide all six elements. The time is always interpreted as UTC.

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

`num_blocks` attribute

```
num_blocks=num
```

Set the number of blocks in the ETFS. If the number of blocks is specified, then the image file will be padded out to that size.

`optional` attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the embedded filesystem. If false, and the host file can't be found, output an error message and exit `mketfs`. The default is true.

`perms` attribute

```
perms=perms_spec
```

Set the access permissions of the file. If specified as a number, the permissions are set to that number (just like the `chmod` command). If specified as an asterisk ("*"), the host file's permissions are used; for an inline file, permissions of 0666 are used. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions.

The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`).

The default *perms_spec* is `*`.

Copyright © 2014, QNX Software Systems Limited

When running on a Windows host, `mketfs` can't get the execute (`x`), setuid ("set user ID"), or setgid ("set group ID") permissions from the file. Use the `perms` attribute to specify these permissions explicitly. You might also have to use the *`uid`* (p. 1227) and *`gid`* (p. 1224) attributes to set the ownership correctly. To determine whether or not a utility needs to have the setuid or setgid permission set, see its entry in the *Utilities Reference*.

`prefix` attribute

`prefix=`*`prefix_spec`*

Set the prefix on the target file names. The default is the empty string.

`search` attribute

`search=`*`path`*`:`*`path`*`:…`

This attribute specifies that `mketfs` should search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the ETFS.

Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

`type` attribute

`type=`*`file_type`*

Sets the type of the files being created in the ETFS. Allowable types are:

- `link` — a symbolic link
- `file` — a regular, everyday file (the default)
- `dir` — a directory.

Specifying `[type=dir]` tells `mketfs` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```


Copyright © 2014, QNX Software Systems Limited

uid attribute

uid=*id_spec*

Set the user ID number for the file. The value of this attribute may be either a number or an asterisk (*). If it's an asterisk, the user ID is taken from the host file; for an inline file, the user ID is the user running mketfs. The default value for this attribute is *.

Examples:

Here's a sample buildfile, my_etfs.bld:

```
# A sample buildfile for mketfs

[cluster_size=1k block_size=64k num_blocks=240]
/home/jgarvey/nto_flash
```

In this example, we've specified a cluster_size of 1 KB, a block_size of 64 KB and a total device size of 240 blocks (which is the default configuration of fs-etfs-ram). The files and subdirectories from the /home/jgarvey/nto_flash directory on the host system are to be recursively copied into the root directory of the ETFS.

To create an ETFS image file using the above buildfile, invoke mketfs as follows:

```
mketfs my_etfs.bld my_image.etfs
```

This creates the my_image.etfs file containing the ETFS filesystem, which can then be copied to the target system as follows:

```
etfsctl -d /dev/etfs2 -S -e -w my_image.etfs -c
```

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mkfatfsimg

Build a FAT filesystem image (QNX)

Syntax:

```
mkfatfsimg [option...] [buildfile] [directory] [outputfile]
mkxfs -t fatfsimg [option...] [buildfile] [directory]
[outputfile]
```

Runs on:

Linux, Microsoft Windows

Options:

-D

Treat undeclared intermediate directories as errors. If there's a target filesystem entry of `/x/y`, and `/x` has never occurred explicitly in the buildfile, the patch file, the input directory, or as a child of a recursively included directory, then `/x` is considered an undeclared intermediate directory.

-d

Display warnings for undeclared intermediate directories.

-l *inputline*

("el") Process *inputline* before interpreting the buildfile. Input lines given to `mkfatfsimg` should be quoted to prevent interpretation by the shell. Multiple -l options are processed in the order specified. This option is especially useful for setting global attributes when the input is a directory only.

-n[n..]

Don't use timestamps in the files. Using the -n option permits identical images in binary format. One n strips timestamps from files that vary from run to run. More than one strips ALL time information, which is necessary on Windows NTFS with daylight saving time.

-p *patchfile*

Apply patching instructions from this file (see "*Patch files* (p. 1232)," below).

-r *root*

Copyright © 2014, QNX Software Systems Limited

Search the default paths in this directory before the default.

-v[v..]

Operate verbosely. Specifying additional v options increases the verbosity. The default is quiet operation.

Description:

The `mkfatfsimg` utility reads a text buildfile and/or a specified directory and produces a binary image file containing a FAT (*`fs-dos.so`* (p. 795)) filesystem created from the given input. You can copy this file to target media at a later stage.

Don't confuse this command with *`mkdosfs`* (p. 1205), which initializes an empty FAT filesystem and is available for Neutrino hosts only.

You specify the input and output on the command line:

buildfile

The filename of the buildfile that describes the contents of the FAT filesystem; use a hyphen (`-`) to specify standard input (the default).

directory

The root of a directory hierarchy to be appended to the file list specified in *buildfile* (if any). The default is no directory.

outputfile

The filename of the image file containing the FAT filesystem; use a hyphen (`-`) to specify standard output (the default). Note that you can specify the output file only if you specified at least either a buildfile or a directory.

If you don't specify either a buildfile or a directory, a buildfile is expected as input from standard input. The output is always an image file; if you don't specify *outputfile*, image-file data will be produced on standard output.

This utility supports long filenames (VFAT).

By default, the FAT type (12, 16, or 32) is derived from the volume size and other geometry parameters. You can use the *`fat`* (p. 1234) attribute to override this.

- The FAT filesystem supports only file sizes less than 4 GB.
- There's only very limited support for permissions in FAT* filesystems. A file or directory can be flagged as "read-only"; `mkfatfsimg` sets this flag when it encounters a file or directory without write permissions for anybody.

Copyright © 2014, QNX Software Systems Limited

- There's no support for hard or symbolic links in FAT filesystems. If you try to add a link to a FAT filesystem image, `mkfatfsimg` issues a warning and skips to the next file.
- FAT filesystems have no concept of ownership. If you use the `uid` or `gid` attribute in the buildfile, it's silently ignored.

Buildfiles

The `mkfatfsimg` command uses the same buildfile grammar as `mkifs`, but supports a different set of attributes. The buildfile is basically just a list of files that you want to be included in the FAT image file when it's built by `mkfatfsimg`. As well as identifying the files to be included, you can specify various attributes that are used to set parameters of the filesystem and the files in it. For example, you can specify the maximum size of the filesystem, or make individual files read-only.

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

  ```
  [attribute] A
  B
  C
  ```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

  ```
  [attribute]
  A
  B
  C
  ```

Attributes provide information about the file following the attribute. They are enclosed in square brackets; when combining attributes (e.g., to specify both the user ID and

Copyright © 2014, QNX Software Systems Limited

the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
[uid=5 gid=5] filename
# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

Boolean attributes

Those prefixed with a plus (+) or minus (-) sign.

Value attributes

Those ending with an equals sign (=) followed by a value. Don't put any spaces around the equals sign.

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example:

```
?+followlink
```

sets the `+followlink` attribute only if `+followlink` or `-followlink` hasn't already been set.

The *file_specification* takes one of the following forms:

path

The file is copied from the host to the location in the image defined by the *prefix* (p. 1236) attribute. If *path* isn't absolute, `mkfatfsimg` looks for it in the locations identified by the *search* (p. 1236) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces ({ })—the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mkfatfsimg` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mkfatfsimg` doesn't interpret a pound sign (#) in an inline file as the beginning

Copyright © 2014, QNX Software Systems Limited

of a comment. The syntax of the inline file depends on what it's used for on the target system.

Closing braces (}) and backslashes (\) in an inline file must be escaped with a backslash.

You can enclose a filename in double quotes (") if it includes spaces or unusual characters.

Patch files

Patch files let you override the user ID, group ID, and permissions of certain files, depending on their location and filename pattern. Patches are applied after all files have been collected (from the buildfile and/or the specified directory). Consequently, patch files can override settings specified in the buildfile.

Patch files must contain only lines of the form:

```
#comment
```

or:

```
type:path:pattern:uid:gid:perms
```

In comment lines, # must be the very first character. The entire line is regarded as a comment and is ignored.

The *type* is either `d` or `f`, optionally followed by `r`. Type `d` patches are applied only to directories, and type `f` patches are applied only to files. An `r` indicates that the patch should be applied recursively within *path*; without `r`, the patch is applied to *path* only.

The *pattern* is a filename pattern that specifies which files to apply the patch to. The *uid* and *gid* must be decimal numbers, while *perms* must be an octal number (see *chmod* (p. 124)). Note that it isn't possible to set only the user ID, group ID, or permissions; for each match, all three are affected. As mentioned above, the FAT filesystem ignores any settings for the user ID and group ID.

Attributes

In `mkfatfsimg` buildfiles, the following attributes are supported:

- *cd=path* (p. 1233)
- *dperms=perm_spec* (p. 1233)
- *fat=type* (p. 1234)
- *filter=filter_spec* (p. 1234)
- *+|-followlink* (p. 1234)
- *media=number* (p. 1234)
- *mtime=time_spec* (p. 1235)

Copyright © 2014, QNX Software Systems Limited

- *num_sectors=size_spec* (p. 1235)
- *+|-optional* (p. 1235)
- *perms=perm_spec* (p. 1235)
- *prefix=path* (p. 1236)
- *search=path[:path...]* (p. 1236)
- *sec_per_clus=number* (p. 1236)
- *sector_size=size_spec* (p. 1236)
- *type=file_type* (p. 1237)
- *vol_lbl=string* (p. 1237)

The following attributes are recognized, but not semantically supported by `mkfatfsimg`:

- `gid=`*id_spec*
- `mountperms=`*perm_spec*
- `uid=`*id_spec*

Since the default values will generate only a 1,44MB FAT12 image, you should explicitly specify at least the image size by specifying `num_sectors`, to make sure that the image produced will be large enough to hold all specified files and that it will be compatible with your specific target device.

An OR-bar indicates that either the first element or the second element must be present, but not both (e.g., `+|-bigendian` means either `+bigendian` or `-bigendian`, but not `+-bigendian`).

cd attribute

`cd=`*path*

Set the current working directory to the specified pathname before attempting to open the host file. The default is the directory from which you invoked `mkfatfsimg`.

dperms attribute

`dperms=`*perm_spec*

Set the access permissions of the directory. If specified as a number, the permissions are set to that number (just like the *chmod* (p. 124) command). If you specify the permissions as an asterisk (`*`), the host directory's permissions are used; for an inline directory, the permissions are obtained from the umask of the user running `mkfatfsimg`. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions. The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

Copyright © 2014, QNX Software Systems Limited

You can include more than one symbolic mode string, separating them with a comma (`,`). The default `dperms_spec` is `*`.

Note that FAT filesystems don't support permissions; they have only a read-only attribute. If the combination of permissions indicates that the directory should not be writeable for anyone, then the read-only attribute is set.

`fat` attribute

`fat=`*`type`*

Set the FAT type of the target filesystem. The *type* must be one of 12, 16, or 32. Note that by defining a specific target FAT type, you may limit the parameter choices so far that no valid filesystem can be constructed. It's usually best to leave this as the default, which is to auto-determine a reasonable FAT type. See the *`sec_per_clus`* (p. 1236) attribute for how this works.

`filter` attribute

`filter=`*`filter_spec`*

Run the host file through the filter program specified, presenting the host file data as standard input to the program and using the standard output from the program as the data to be placed into the FAT filesystem. Default is no filter. You can specify a `filter_spec` of none. This is useful if you need to override a global filter specification.

`followlink` attribute (boolean)

`[+|-followlink]`*`target_path=host_path`*

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mkfatfsimg` resolves the symbolic link and includes its target file or directory. You shouldn't change this behavior, since FAT filesystems don't support symbolic links. If you specify `-followlink`, and `mkfatfsimg` encounters a symbolic link, it issues a warning message and skips to the next file.

`gid` attribute

`gid=`*`id_spec`*

This attribute is supposed to set the group ID number for the file. Since FAT filesystems have no provision to store file ownership information, this attribute is silently disregarded.

`media` attribute

`media=`*`number`*

Specify a media indicator code. The default value is `0xF8`.

Copyright © 2014, QNX Software Systems Limited

mountperms attribute

This attribute is supposed to set the access permissions for the filesystem mountpoint. Since the FAT filesystem has no provision to store mount information, this attribute is silently disregarded.

mtime attribute

```
mtime=time_spec
```

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (*), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```

  You must provide all six elements. The time is always interpreted as UTC.

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

num_sectors attribute

```
num_sectors=size_spec
```

Set the number of 512-byte sectors on the volume. The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). The default value is 2880.

optional attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the embedded filesystem. If false, and the host file can't be found, output an error message and exit `mkfatfsimg`. The default is true.

perms attribute

```
perms=perm_spec
```

Set the access permissions of the file. If specified as a number, the permissions are set to that number (just like the *chmod* (p. 124) command). If specified as an asterisk (*), the host file's permissions are used; for an inline file, permissions of 0666 are used. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions. The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

Copyright © 2014, QNX Software Systems Limited

You can include more than one symbolic mode string, separating them with a comma (`,`). The default *perms_spec* is `*`.

Note that FAT filesystems don't support permissions; they have only a read-only attribute. If the combination of permissions indicates that the file should not be writeable for anyone, then the read-only attribute is set.

`prefix` attribute

`prefix=`*`path`*

Set the prefix on the target file names. The default is the empty string.

`search` attribute

`search=`*`path`*`[:`*`path...`*`]`

This attribute specifies that `mkfatfsimg` should search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the FAT filesystem. Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

`sec_per_clus` attribute

`sec_per_clus=`*`number`*

Define the number of sectors assigned per each cluster. This must be one of 1, 2, 4, 8, 16, 32, or 64. Note that while "officially" allowed, you can't choose a value of 128 because it would necessarily result in a cluster size > 32K and would likely be incompatible with many systems.

The default is to auto-detect a reasonable sectors/cluster ratio. The built-in algorithm tries to use as few sectors per cluster as possible and use the smallest-possible FAT type. This strategy is meant to minimize allocation overhead (the average over-allocation is 1/2 cluster per each file). Of course, small clusters will go along with large FATs which also take up some space. If you know that your filesystem will contain only few files, you may choose a larger sectors/cluster ratio to keep the FATs smaller.

`sector_size` attribute

`sector_size=`*`size_spec`*

Specify the sector size of the filesystem's target device. The *size_spec* is an integer, optionally followed by `k` (the case doesn't matter). The default value is 512; valid values are 512, 1k, 2k, 4k, 8k, 16k, and 32k. Note that any sector size greater than 4k may be incompatible with some operating systems and will cause a warning to be issued.

Copyright © 2014, QNX Software Systems Limited

type attribute

`type=file_type`

Sets the type of the files being created in the FAT filesystem. Allowable types are:

- `file`—a regular, everyday file (the default)
- `dir`—a directory

Specifying `[type=dir]` tells `mkfatfsimg` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```

Note that `link` is also an accepted type name, but since FAT filesystems don't support symbolic links, a warning will be issued and all attempts to create symbolic links will be skipped.

uid attribute

`uid=id_spec`

This attribute is supposed to set the user ID number for the file. Since FAT filesystems have no provision to store file ownership information, this attribute is silently disregarded.

vol_lbl attribute

`vol_lbl=string`

Set the volume label to the given string. The default is no label.

Examples:

Here's a sample buildfile, `my_fatfs.bld`:

```
# A sample buildfile for mkfatfsimg
[num_sectors=512k]
/home/thaupt
```

In this example, we've specified a sector count of 512*1024, which relates to an image size of 128 MB. The files and subdirectories from the `/home/thaupt` directory on the host system are to be recursively copied into the root directory of the FAT

Copyright © 2014, QNX Software Systems Limited

filesystem. To create a FAT filesystem image file using above buildfile, invoke `mkfatfsimg` as follows:

```
mkfatfsimg my_fatfs.bld my_fatfs.img
```

This creates the `my_fatfs.img` file containing the FAT filesystem, which you could then copy to a target system's hard-disk partition as follows:

```
dd if=my_fatfs.img of=/dev/hd0t7 count=524288
```

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mkfifo

Make FIFO special files (POSIX)

Syntax:

```
mkfifo [-p] [-m mode] file...
```

Runs on:

QNX Neutrino

Options:

-m *mode*

When creating the FIFO special file, set the file permission bits of the new file to the specified *mode* value.

The *mode* option argument is a *symbolic_mode* string, as defined for the *chmod* (p. 124) utility. In the *symbolic_mode* strings, the *op* characters + and - are interpreted relative to the default file mode for that file type, as follows:

+

Add permissions to the default mode.

-

Delete permissions from the default mode.

=

Assign permissions.

-p

Create directories in the FIFOs pathname, if required.

file

The pathname at which a FIFO special file is to be created.

Description:

The `mkfifo` utility creates the FIFO special files specified by the *file* operands in the order they're specified.

Copyright © 2014, QNX Software Systems Limited

To create a FIFO in a directory, you must have write permission for that directory or be logged in as `root`.

The default file mode for FIFO files is `a=rw` (666) with selected permissions removed in accordance with the file mode creation mask (see *umask* (p. 2005)). For intermediate pathname components created by `mkfifo`, the mode is the default modified by `u+wx` so that any subdirectories and the FIFO file can always be created regardless of the file mode creation mask. If you wish to assign different ultimate permissions for the intermediate directories, you can change these permissions afterward with the *chmod* (p. 124) utility.

Exit status:

0

Success.

> 0

An error occurred.

Caveats:

If the `mkfifo` utility is terminated by a signal, some of the specified FIFO special files or intermediate directories might have already been created, and may not be automatically removed.

Copyright © 2014, QNX Software Systems Limited

mkifs

Build an OS image filesystem (QNX)

Syntax:

```
mkifs [-a suffix] [-l inputline] [-n[n]] [-r rootdir]
      [-s section] [-v] [buildfile [imagefile]]
```

Runs on:

Linux, Microsoft Windows

Options:

-a *suffix*

Append a suffix to symbol files generated via [+keeplinked].

-l *inputline*

("el") Process *inputline* before interpretation of the buildfile begins. Input lines given to `mkifs` must be quoted to prevent interpretation by the shell (especially since `mkifs` input lines often contain spaces). Multiple -l options are processed in the order specified. No default.

-n[n]

Force the modification times of all inline files to be 0. If you specify -nn, `mkifs` sets the modification times of *all* files to 0.

When `mkifs` adds files to an IFS image, it uses the timestamp information from the file on the host machine. If `mkifs` is creating an inline file (which doesn't exist on the host machine), it has to generate its own timestamp information. By default, it's the time that the image is generated.

This results in different checksum values for two identical builds (because the file's creation or modification times are different). If you use -n, the checksum value is the same on all identical builds.

The -nn option addresses a quirk in NTFS relating to daylight savings time. This option forces the modification time for *all* files in the IFS image to be set to 0. This ensures that subsequent builds of the same IFS image have the same checksum.

-r *rootdir*

Copyright © 2014, QNX Software Systems Limited

Search the default paths in the *rootdir* directory before searching them in the default location. Normally, `mkifs` searches the default paths in the sequence shown in ***MKIFS_PATH*** (p. ?) below. If you specify the -r option, `mkifs` first searches the same paths prefixed with the *rootdir* variable instead of `${`***QNX_TARGET***`}`, like this:

1. *rootdir*`/${`***PROCESSOR***`}/sbin`
2. all other default paths, similarly prefixed with *rootdir*
3. `${`***QNX_TARGET***`}/${`***PROCESSOR***`}/sbin`
4. all other normal default paths prefixed with `${`***QNX_TARGET***`}`

The structure of the directory paths under *rootdir* must be identical to that of the default paths under `${`***QNX_TARGET***`}`, but *rootdir* itself may be any path you choose. For example, if you wanted to include `/dev/armle-v7/sbin/devb-eide`, you would specify the option like this:

```
-r /dev
```

Notice that you don't include `${`***PROCESSOR***`}` in *rootdir*.

If you set ***MKIFS_PATH***, `mkifs` ignores the -r option.

-s *section*

Don't strip the named section from ELF executables when creating an IFS image. You can use this option more than once to specify additional sections.

By default, `mkifs` doesn't strip the `QNX_usage` (usage message), and `QNX_info` (build properties) sections.

-v

Operate verbosely. Specifying additional -v options increases verbosity. Default is quiet operation.

Description:

The `mkifs` utility is used to create an OS image filesystem from a buildfile specification.

Don't confirm this command with *`mkefs`* (p. 1209), which builds an *embedded* filesystem, or *`mketfs`* (p. 1219), which builds an embedded transaction filesystem (ETFS).

You can specify these files on the command line:

Copyright © 2014, QNX Software Systems Limited

buildfile

The input buildfile that `mkifs` is to construct an image from; use `-` to specify standard input (the default).

imagefile

The file to contain the image that `mkifs` builds; use `-` to specify standard output (the default). Note that you can specify the *imagefile* only if you've specified the *buildfile*.

Buildfiles

The buildfile uses the same grammar as the *mkefs* (p. 1209) command, but supports different attributes.

The buildfile specifies a list of files of various types; these files are placed by `mkifs` into the output image. As well as the files to be included, you can specify various attributes that are used to set parameters of the files or the image as a whole.

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

  ```
  [attribute] A
  B
  C
  ```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

  ```
  [attribute]
  A
  B
  C
  ```

Copyright © 2014, QNX Software Systems Limited

Enclose the attributes in square brackets; when combining attributes (e.g. to specify both the user ID *and* the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
[uid=5 gid=5] filename

# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

boolean attributes

Those prefixed with a plus (“+”) or minus (“-”) sign.

value attributes

Those ending with an equals sign (“=”) followed by a value. Don't put any spaces around the equals sign.

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example, `?+bigendian` sets the `+bigendian` attribute only if `+bigendian` or `-bigendian` hasn't already been set.

The *file_specification* takes one of the following forms:

path

The file is copied from the host to the location in the image defined by the *`prefix`* (p. 1253) attribute. If *path* isn't absolute, `mkifs` looks for it in the locations identified by the *`search`* (p. 1256) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces (`{ }`) — the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mkifs` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mkifs` doesn't interpret a pound sign (#) in an inline file as the beginning of a comment. The syntax of the inline file depends on what it's used for on the target system.

Copyright © 2014, QNX Software Systems Limited

Closing braces (}) and backslashes (\) in an inline file must be escaped with a backslash.

Either class of file may be preceded by zero or more attributes. These attributes are used to specify the characteristics of the file (e.g. the user ID that is to own the file on the target system, the type of file, etc.).

By default, `mkifs` strips debugging information from executable files that you include in the image. Doing this helps to reduce the size of the image. To keep this information, specify the *+raw* (p. 1255) attribute.

By default, `mkifs` doesn't strip the `QNX_usage` (usage message), and `QNX_info` (build properties) sections. You can use the -s option to specify additional sections not to be stripped.

You can enclose a filename in double quotes ("") if it includes spaces or unusual characters.

Attributes

The `mkifs` command supports the following attributes:

- *+|-autolink* (p. 1246)
- *+|-big_pages* (p. 1247)
- *+|-bigendian* (p. 1247)
- *cd=path* (p. 1247)
- *chain=addr* (p. 1247)
- *code=uip_spec* (p. 1247)
- *compress* (p. 1247)
- *data=uip_spec* (p. 1248)
- *dperms=perm_spec* (p. 1248)
- *filter=filter_spec* (p. 1248)
- *+/-followlink* (p. 1248)
- *gid=id_spec* (p. 1249)
- *image=addr_space_spec* (p. 1249)
- *+|-keeplinked* (p. 1250)
- *linker= [linker_id_spec] linker_spec* (p. 1250)
- *module=module_name* (p. 1251)
- *mtime=time_spec* (p. 1251)
- *+|-optional* (p. 1252)
- *+|-page_align* (p. 1252)
- *pagesizes=size[,size]...* (p. 1252)

Copyright © 2014, QNX Software Systems Limited

- *perms=perm_spec* (p. 1252)
- *phys_align=size* (p. 1253)
- *physical=boot_spec* (p. 1253)
- *prefix=prefix_spec* (p. 1253)
- *ram=addr_space_spec* (p. 1254)
- *+|-raw* (p. 1255)
- *+|-script* (p. 1255)
- *search=path:path:...* (p. 1256)
- *type=file_type* (p. 1256)
- *uid=id_spec* (p. 1256)
- *virtual=[cpu_name,]bootfile_name [filter_args]* (p. 1257)

An OR-bar indicates that either the first or second element must be present, but not both (e.g. +|- `bigendian` means either `+bigendian` or `-bigendian`, but not `+-bigendian`).

`autolink` attribute (boolean)

```
+|-autolink
```

If the `autolink` attribute is on (which it is by default), when `mkifs` detects that it's processing a shared object, it looks inside the image for the SONAME (specified by the linker -h option). This is typically the shared object name, including the version number (e.g. `libc.so.1`). The `mkifs` command puts the file into the image filesystem under the name with the version number and makes the name without the version number into a symbolic link to the file. For example, specifying:

```
libc.so
```

in the buildfile makes `libc.so.1` the name of the file and `libc.so` a symlink to it. Specifying:

```
libc.so.1
```

in the buildfile gives the same results. You end up with the name with and without the version number in the image filesystem no matter which one you specify in the buildfile.

If the name that would be used as the symbolic link is already specified somewhere else in the buildfile, the symbolic link isn't created. For example:

```
libc.so.1
libc.so.2
[type=link] libc.so=libc.so.2
```

ensures that `libc.so` is pointing at the "proper" version of the library.

You can disable this feature by specifying the `-autolink` attribute.

Copyright © 2014, QNX Software Systems Limited

+|-big_pages attribute (boolean)

This attribute makes `mkifs` attempt to align binaries and shared libraries at the appropriate address, based on the size of the UIP text. You can use the *pagesizes* (p. 1252) attribute to indicate the page sizes that the underlying hardware supports. You can specify these attributes in the buildfile or in the *bootfile* (p. 1262). The `big_pages` attribute is off by default.

A *phys_align* (p. 1253) attribute overrides the `big_pages` alignment hint. For example:

```
[+big_pages pagesizes=4k,64k]
libc.so
[-big_pages]a_binary_that_doesnt_want_bigpages

[phys_align=1m]explicit_override_for_this_file
```

bigendian attribute (boolean)

```
+|-bigendian
```

Set the byte order for the image filesystem to either big (via `+bigendian`) or little (via `-bigendian`) endian. This option doesn't normally need to be specified when building a bootable image, since the bootfile provides the required byte order. If you aren't building a bootable filesystem, or the bootfile doesn't say which byte order to use, `mkifs` uses the host system's byte order in building the image filesystem.

cd attribute

```
cd=path
```

Set the current working directory to the specified pathname before attempting to open the host file. Default is the directory from which `mkifs` was invoked.

chain attribute

```
chain=addr
```

Set the address at which the operating system will find the next image filesystem. Default is none.

code attribute

```
code=uip_spec
```

Set whether the code segment of an executable is used directly from the image filesystem (`uip` or `u`) or copied (`copy` or `c`) when invoked. The default is to use the code segment in place (`uip`). For more information, see "*Notes on XIP versus copy* (p. 1269)," below.

compress attribute

```
+|-compress
compress=algorithm
```

Copyright © 2014, QNX Software Systems Limited

Set whether the image is compressed. The default is false.

The first (boolean) form turns compression on or off. The algorithm is the default, UCL8.

The second form (note there's no leading + or - sign) turns compression on and specifies the algorithm by number:

- 1 — ZLIB
- 2 — LZO
- 3 — UCL8 (the default)

`data` attribute

```
data=uip_spec
```

Set whether the data segment of an executable is used directly from the image filesystem (`uip` or `u`) or copied (`copy` or `c`) when invoked. The default is to use the data segment in place (`uip`). For more information, see "*Notes on XIP versus copy* (p. 1269)," below.

`dperms` attribute

```
dperms=perm_spec
```

Set the access permissions of the directory. See the *`perms`* (p. 1252) attribute for more information.

`filter` attribute

```
filter=filter_spec
```

Run the host file through the filter program specified, presenting the host file data as standard input to the program, and use the standard output from the program as the data to be placed into the image filesystem. Default is no filter.

To illustrate the use of a filter, consider storing a compressed file in the image filesystem, where the file exists in its uncompressed form on the host filesystem:

```
[filter="compress"] data.Z = data
```

This runs `compress` from a shell, passing it the contents of `data` as standard input. The `compress` command runs and generates the compressed version of its standard input on its standard output. The standard output is then placed into the image filesystem as the `data.Z` file.

You can specify a *filter_spec* of `none`. This is useful if you need to override a global filter specification.

`followlink` attribute (boolean)

```
[+|-followlink]target_path=host_path
```


Copyright © 2014, QNX Software Systems Limited

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mkifs` resolves the symbolic link and includes its target file or directory. If you specify `-followlink`, `mkifs` includes the symbolic link itself in the image filesystem. It's up to you to include in the image whatever the link points to.

`gid` attribute

`gid=`*`id_spec`*

Set the group ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the group ID is taken from the host file; for an inline file, the group ID is the group of the user running `mkifs`. The default value for this attribute is `*`.

`image` attribute

`image=`*`addr_space_spec`*

Set the base and size limits for the image filesystem. The format for this attribute consists of an optional starting address, followed by zero or more parameters for sizing the address space. You can use a case-insensitive suffix of `k`, `m`, or `g` on the addresses and sizes.

The starting address is the base address of the image, and matters only when building a bootable image. Its default depends on the bootfile selected. For example, on an x86 using the `bios.boot` file, the image address begins at 4 MB.

-end_addr

A dash followed by a number represents an ending address, the last allowable address in the image. If the output image exceeds this address, an error is reported. The default is no limit.

,maxsize

A comma followed by a number represents the maximum allowed size of the image. If the output image becomes larger than this value, an error is reported. The default is no limit. The maximum image size depends on your configuration; for example, it may be limited on an x86 system with a BIOS.

=totalsize

An equals sign followed by a number represents the total size that the output image is padded out to. The default is no padding.

%align

Copyright © 2014, QNX Software Systems Limited

A percent sign followed by a number represents the alignment value used for the image. The output image size is padded out to a multiple of this value. The default is 4.

You have to specify both `image` and *`ram`* (p. 1254) file attributes if you want to create the image in ROM/FLASH; otherwise the process manager assumes that the image is in RAM. For more information, see "*Notes on XIP versus copy* (p. 1269)," below.

`keeplinked` attribute (boolean)

```
+|-keeplinked
```

If true, and `mkifs` has to run a linker to position the executable within the image filesystem, the output file from the link is the basename of the host path with `.sym` appended. For example, if the host name is `../foo/bar`, the output name is `bar.sym`. If false, a generated temporary name is used for the output file and it's deleted after `mkifs` has run. The default is false.

`linker` attribute

```
linker=[linker_id_spec]linker_spec
```

When building a bootable image, `mkifs` sometimes needs to run a linker on relocatable objects to position them within the image. This option lets you specify *printf*-like macro expansions to tell `mkifs` how to generate the linker command line (see "*Linker Specification* (p. 1265)," below for details).

You don't normally need to specify this option, since `mkifs` or a bootfile provides a default. You can use different linkers for different types of ELF files.

The attribute value consists of an optional linker ID specification and a linker specification. The linker ID specification, if present, consists of:

1. An opening parenthesis, `(`.
2. A list of comma-separated numbers giving the allowable ELF machine numbers (`EM_*` constants from the include file `<sys/elf.h>`) for the linker specification. Terminate the list of machine numbers with a semicolon.
3. A list of comma-separated numbers, giving the list of acceptable ELF file types (`ET_*` constants, from `<sys/elf.h>`). Terminate this list with a semicolon.
4. A comma-separated list of numbers giving ELF program segment types (`PT_*` constants, also from `<sys/elf.h>`).
5. A closing parenthesis, `)`.

If the ID specification is present, the linker specification is used only if the machine number of the ELF input file matches one of the given numbers, *and* the ELF file type

Copyright © 2014, QNX Software Systems Limited

of input file matches one of the given numbers *and* at least one of the program segment types in the input file matches one of the given numbers:

- If the machine number list is empty, any machine number type in the input file is acceptable.
- If the program segment number list is empty, any program segment number types in the input file are acceptable.
- If the ELF file type number list is empty, ET_REL is assumed.

module attribute

```
module=module_name
```

Use this attribute to add optional modules to `procnto`.

For example, in order to use the adaptive partitioning scheduler, you must rebuild your OS images with the option `[module=aps]` added to the PATH= statement of your buildfile:

```
[module=aps] PATH=/proc/boot ./procnto -vv
```

Y ou can now create partitions and launch applications within a particular partition for the adaptive partitioning scheduler.

For information on creating a partition, see “Creating partitions” in the Setting Up and Using the Adaptive Partitioning Scheduler chapter of the Adaptive Partitioning *User's Guide*.

For information on launching applications within a particular partition, see “Launch processes in partitions” in the Setting Up and Using the Adaptive Partitioning Scheduler chapter of the Adaptive Partitioning *User's Guide*.

This attribute was added in the QNX Neutrino Core OS 6.3.2.

mtime attribute

```
mtime=time_spec
```

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (*), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```

  You must provide all six elements. The time is always interpreted as UTC.

Copyright © 2014, QNX Software Systems Limited

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

`optional` attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the image filesystem. If false, and the host file can't be found, output an error message and exit `mkifs`. The default is true. You can't set this attribute to true for bootstrap executables (see the *`virtual` attribute* (p. 1257) for more information).

`page_align` attribute (boolean)

```
+|-page_align
```

If true, align the file on a page boundary. The `mkifs` utility always aligns executables and shared objects on page boundaries, so this attribute has an effect only on data files and files that you specify *`+raw`* (p. 1255) for.

`pagesizes` attribute

```
pagesizes=size[,size]...
```

This attribute defines the page sizes that the underlying hardware supports, for use with the *`big_pages`* (p. 1247) attribute. The sizes can be in any order, and can include a case-insensitive suffix of `k`, `m`, or `g`.

You can specify these attributes in the buildfile or in the *bootfile* (p. 1262).

`perms` attribute

```
perms=perm_spec
```

Set the access permissions of the file. The *perm_spec* can be one of the following:

- a number (just as with the `chmod` command)
- an asterisk (`*`) to use the host file's permissions (or 0666 for inline files)
- a symbolic mode string to delete, add, or set permissions. This string is a subset of `chmod`'s and consists of:

  1. a combination of `u`, `g`, `o`, and `a`
  2. one of `-`, `=`, or `+`
  3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

  You can include more than one symbolic mode string, separating them with a comma (`,`).

The default is `*`.

When running on a Windows host, `mkifs` can't get the execute (`x`), setuid ("set user ID"), or setgid ("set group ID") permissions from the file. Use the

Copyright © 2014, QNX Software Systems Limited

perms attribute to specify these permissions explicitly. You might also have to use the *uid* (p. 1256) and *gid* (p. 1249) attributes to set the ownership correctly. To determine whether or not a utility needs to have the setuid or setgid permission set, see its entry in the *Utilities Reference*.

ELF executables and shared objects are automatically marked as executable (unless you specify [+raw]).

---

phys_align attribute

```
phys_align=size[,group]
```

The phys_align attribute lets you align IFS objects on specific physical address boundaries to take advantage of large pages. The size is an integer, optionally followed by k, m, or g in lower- or uppercase. This attribute overrides the *+|-big_pages* (p. 1247) attribute.

For example, to align a file on a 64 KB boundary, potentially allowing the use of 64 KB pages to map 64 KB chunks of the file, specify:

```
[phys_align=64k] some_executable
```

You can use the optional group option to group shared objects together based on an alignment size. For example:

```
[phys_align=16M,group]
first.so
second.so
third.so
[phys_align=0] # ends alignment
```

In this example, first.so is aligned to 16 MB, and each successive shared object either completely fits within that same 16 MB page, or is bumped to the next 16 MB boundary.

physical attribute

```
physical=[cpu_name,]boot_filename [filter_args]
```

This attribute indicates that a bootable filesystem is being built. You can specify it only once in a buildfile. The image will be run in physical memory mode.

The physical attribute isn't currently implemented; use virtual.

For more information, see the *virtual attribute* (p. 1257).

prefix attribute

```
prefix=prefix_spec
```

Copyright © 2014, QNX Software Systems Limited

Set the prefix for the target file names. Default is `proc/boot` when building a bootable image, and the empty string when not.

`ram` attribute

`ram=`*`addr_space_spec`*

Set base and size limits for the read-write memory required by executables in the image filesystem. This attribute consists of an optional starting address, followed by zero or more parameters for sizing the address space. You can use a case-insensitive suffix of `k`, `m`, or `g` on the addresses and sizes.

You have to specify both *`image`* (p. 1249) and `ram` file attributes if you want to create the image in ROM/FLASH; otherwise the process manager assumes that the image is in RAM. For more information, see "*Notes on XIP versus copy* (p. 1269)," below.

You need to specify this attribute if the actual image is going to be stored on a read-only device such as ROM or flash memory. Use the *`image`* (p. 1249) attribute to specify the location.

The starting address specifies the base address of the RAM, and matters only when building a bootable image. The default depends on the bootfile you select.

-end_addr

A dash followed by a number represents an ending address, the last allowable address for RAM. If the RAM usage exceeds this address, an error is reported. The default is no limit.

,maxsize

A comma followed by a number represents the maximum allowed size of the RAM. If the output image requires more RAM than this value, an error is reported. The default is no limit. The maximum RAM size depends on your configuration.

=totalsize

An equals sign followed by a number represents the total size that the RAM usage is padded out to. The default is no padding.

%align

A percent sign followed by a number represents the alignment value used for the RAM. The RAM size is padded out to a multiple of this value. The default is 4.

Copyright © 2014, QNX Software Systems Limited

For information about how everything interacts, see "*Notes on XIP versus copy* (p. 1269)," below.

`raw` attribute (boolean)

`+|-raw`

If the `raw` attribute is false (the default), `mkifs` strips debugging information from executable files.

If you specify `+raw` for a file, the file is treated as a data file, even if it would normally be treated as an executable and relocated.

Don't specify the `+raw` attribute for shared objects; it prevents them from being shared.

If you use the default attribute (`-raw`) and you specify that the data segment is to be used in place, the file's sticky bit will not be set. This identifies the executable to the QNX Neutrino process manager, which will prevent the program from running more than once, avoiding the possibility of running a program with corrupted static data.

Here's a fragment from a buildfile that demonstrates the use and scope of the `raw` attribute:

```
…
[+raw]              # Don't strip debugging information
my_app1

[-raw] esh          # Only esh is affected

[-raw]              # Turn off +raw, since shared objects
libphrender.so      # can't be shared if +raw is enabled
libph.so

[+raw] my_app2      # We want debugging information for this
                    # file only.  The -raw flag is
                    # still in effect for other files.
libc.so             # Still affected by -raw flag
…
```

If you use Windows tools to build your image, files specified as `+raw` won't have executable permission when the image is booted. This is because the win32 filesystem can't set a file's executable bit. To make such files executable, specify `perms=+x` when using `+raw` in the buildfile.

`script` attribute (boolean)

`+|-script`

If true, the host file is opened and processed as a script file after the process manager has initialized itself. Each line is parsed as a command line to be run. If multiple files are marked with `+script`, they're merged sequentially into a single file in the image filesystem; the file's name is the first script filename in the buildfile. The filenames

Copyright © 2014, QNX Software Systems Limited

for the subsequent script files are ignored, but they must be unique. See "*Script Files* (p. 1258)," below for more details on the command-line syntax.

`search` attribute

`search=`*`path`*`:`*`path`*`:…`

This attribute specifies that `mkifs` is to search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the image filesystem. The default is the contents of the ***`MKIFS_PATH`*** (p. ?) environment variable.

Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

`type` attribute

`type=`*`file_type`*

Set the type of the files being created in the image filesystem. Allowable types are:

- `link` — a symbolic link
- `fifo` — a named pipe
- `file` — a regular, everyday file (the default)
- `dir` — a directory

Specifying `[type=dir]` tells `mkifs` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```

`uid` attribute

`uid=`*`id_spec`*

Set the user ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the user ID is taken from the host file; for an inline file, the user ID is the user running `mkifs`. The default value for this attribute is `*`.

Copyright © 2014, QNX Software Systems Limited

`virtual` attribute

```
virtual=[cpu_name,]bootfile_name [filter_args]
```

This attribute specifies that a virtual address system is being built.

If there's a comma (`,`) or slash (`/`) in the value, the string in front of it is taken to be the CPU type of the target system. If you don't specify a CPU type, `mkifs` uses the host system's CPU type. The ***PROCESSOR*** environment variable is set to that string (which affects the ***MKIFS_PATH*** search path for host files).

The characters after the comma or slash (or the equal sign for the attribute if there's no comma or slash) up to the first blank character are taken to be the name of the bootfile. The suffix `.boot` is appended to the given name and ***MKIFS_PATH*** is searched for the file. The default bootfiles are in `${`***QNX_TARGET***`}/${`***PROCESSOR***`}/boot/sys`. The bootfiles vary from one processor to another, but these are the main ones:

`binary.boot`

Create a simple binary image (without the jump instruction that `raw.boot` adds). If you build a binary image, and you want to load it with U-Boot (or some other bootloader), you have to execute `mkifs -vvvv buildfile imagefile`, so that you can see what the actual entry address is, and then pass that entry address to the bootloader when you start the image. If you modify the startup code, the entry address may change, so you have to obtain it every time. With a raw image, you can just have the bootloader jump to the same address that you downloaded the image to.

`bios.boot`

Create an image that's suitable for machines with a BIOS. Information that's gathered from the BIOS is put into the startup headers.

`bios16m.boot`

Don't check memory above 16 MB. Use this bootfile with older BIOSs that have trouble with such memory.

`bios_nokbd.boot`

Don't do anything with the keyboard controller. Use this bootfile if the BIOS doesn't properly support or emulate the legacy functionality of keyboard controllers at address `0x64`.

Starting in QNX Neutrino 6.5.0, `bios.boot` contains a test to determine whether or not it should behave like `bios_nokbd.boot`; it should work for the majority of platforms that previously required `bios_nokbd.boot`.

Copyright © 2014, QNX Software Systems Limited

elf.boot

Create an image that looks like an ELF executable.

nobios.boot

Create an image that's suitable for machines that don't have a BIOS.

openbios.boot

Create an image for IBM's OpenBIOS.

raw.boot

Create a binary image with an instruction sequence at its beginning to jump to the offset of *startup_vaddr* within the startup header. The advantage is that when you download a raw image to memory using a bootloader, you can then instruct it to run right at the beginning of the image, rather than having to figure out what the actual *startup_vaddr* is each time you modify the startup code.

srec.boot

Create an image in S-record format.

For more details on the contents of the file, see "*Bootfile* (p. 1262)," below.

Any characters in the attribute value following a blank are used as arguments to any image filter command specified by the bootfile, like this:

```
[virtual="x86,srec -b"] boot = {
```

If the value of the `virtual` attribute includes a space, put quotation marks around the string. Otherwise, `mkifs` will try to interpret the value as an additional buildfile attribute placed in the same set of square brackets.

The contents of the host file that this attribute applies to are parsed to discover the bootstrap executables used to bring up the system. Each line identifies one bootstrap executable:

- The first executable must be the QNX Neutrino startup executable (`startup-*`) that's appropriate for the target system. For more information, see *`startup-*`* (p. 1729).
- The last must be `procnto`. Specifying the ***PATH*** and ***LD_LIBRARY_PATH*** environment variables before `procnto` sets their default values in the OS image.

Script files

As mentioned above, by specifying the `[+script]` attribute, you're telling `mkifs` that the specified file is a *script file*, a sequence of commands to be executed when the process manager has completed its startup.

Copyright © 2014, QNX Software Systems Limited

In order to run a command, its executable must be available when the script is executed. You can add the executable to the image, or get it from a filesystem that's started before the executable is required. The latter approach results in a smaller image.

The bootfile typically sets the _CS_PATH configuration string, and might set _CS_LIBPATH. You can set environment variables, such as ***PATH*** and ***LD_LIBRARY_PATH***, in a script file.

Script files, for the most part, look just like regular shell scripts, except that:

- there are special modifiers that you can place *before* the actual commands to run
- some commands are considered to be built-in
- `mkifs` preparses the script file contents before placing them into the image.

The script file consists of one or more lines, with each line having the following syntax:

```
[modifiers] [command_line [&]]
```

The *modifiers* consist of a list, enclosed in square brackets, of blank-separated items that modify how QNX Neutrino runs the specified *command_line*. If there's a command line following the modifiers, the modifiers affect only that one command line. If there's no command line, the modifiers affect all subsequent command lines.

Startup scripts support foreground and background processes. Just as in the shell, specify an ampersand (&) on the command line to make the program run in the background. If you run a program in the foreground, and it doesn't exit, then the rest of the script is never executed, and the system might not become fully operational.

The modifiers are described below, and include:

- *argv0=value* (p. 1259)
- *cpu=number* (p. 1260)
- *external (boolean)* (p. 1260)
- *pri=priority[sched_policy]* (p. 1261)
- *sched_aps=partition_name* (p. 1261)
- *session (boolean)* (p. 1262)

Those marked as “boolean” accept a plus (+) or minus (-) character to enable or disable the effect; the others accept a parameter.

argv0 modifier

Sets the *argv*[0] element of the command argument entry. By default, this is the same as the command name. This option is typically used to simulate invoking a command

Copyright © 2014, QNX Software Systems Limited

via a different name; the classical example is the `compress` command, which can be invoked as `uncompress`:

```
[argv0=uncompress] compress filename.Z
```

cpu modifier

Specifies the CPU on which to launch the following process (or, if the attribute is used alone on a line without a command, sets the default CPU for all following processes). This modifier is useful for setting up bound multiprocessing (BMP). Specify the CPU as a zero-based processor number:

```
[cpu=0] my_program
```

A value of * allows the processes to run on all processors:

```
[cpu=*] my_program
```

At boot time, if there isn't a processor with the given index, a warning message is displayed, and the command is launched without any runmask restriction.

Due to a limitation in the boot image records, this syntax allows only the specification of a single CPU and not a more generic runmask. Use the *on* (p. 1417) utility to spawn a process within a fully specified runmask.

`external` modifier (boolean)

Ordinarily, `mkifs` recognizes certain commands as *internal commands*, ones that aren't loaded from the host's filesystem, but are understood directly by `mkifs`. These commands are:

`display_msg` *message*

Causes the message immediately following the `display_msg` command to be output. This is useful during startup diagnostics; often used as a checkpoint.

`procmgr_symlink`

Equivalent to `ln -P`, except that you don't need to have *ln* (p. 1114) present.

`reopen` [*filename*]

Causes standard input, standard output, and standard error to be redirected to the specified filename. Also causes the interpretation of the script file to suspend temporarily until a *stat()* on the specified pathname succeeds. The default filename is `/dev/console`.

`waitfor` *pathname* [*wait_time*]

Causes interpretation of the script file to suspend temporarily until a *stat()* on the specified pathname succeeds. Often used for synchronization, to

Copyright © 2014, QNX Software Systems Limited

allow a resource manager to perform its startup functionality, and *then* for the process manager to proceed with the further interpretation of the script file.

The optional *wait_time* specifies the maximum number of seconds to wait for the file to appear. It can include one decimal place to specify tenths of a second. The default is 5.0 seconds.

The `+external` modifier instructs `mkifs` to search for the specified command on the host filesystem, rather than assume the internal meaning for the command. The default is `-external`.

Using `+external` is a dubious practice. Specifying an `external` modifier on a command that isn't an internal command is redundant.

`pri` modifier

Lets you specify the command's priority and optionally the scheduling policy. The `pri` modifier accepts a numeric priority, optionally followed by one of the letters:

`f`

FIFO scheduling policy.

`r`

Round-robin scheduling policy.

`o`

Other scheduling policy (currently maps to round-robin).

See the *System Architecture* guide for a description of the various priority levels and scheduling algorithms.

For example, to start up the console driver, *`devc-con`* (p. 265) at priority 20, with FIFO scheduling, specify:

```
[pri=20f] devc-con -n9 &
```

The default priority and policy are specified by *`procnto`* (p. 1586), and could change in future versions of the QNX Neutrino RTOS. If the priority and scheduling policy of the processes are important to you, be sure to use the `pri=` modifier.

`sched_aps` modifier

Copyright © 2014, QNX Software Systems Limited

Launch the process (or, if the attribute is used alone on a line without a command, all following processes) in the adaptive partition with the specified name:

```
sched_aps=partition_name
```

For example:

```
[+sessionpri=35 sched_aps=DebugReserve] ksh &
```

launches a high-priority shell in the `DebugReserve` partition.

In order to use adaptive partitioning, you must also do the following in your buildfile:

- Specify `[module=aps]` in the line that starts `procnto`.
- Create the partition with a `sched_aps` command in the startup script (before launching any commands in the partition):

  ```
  sched_aps name budget
  ```

For more information, see the Adaptive Partitioning *User's Guide*.

`session` modifier (boolean)

If `+session` is specified, make the process a session leader (as per POSIX), and make the process's *stdin* the controlling terminal (i.e. direct **Ctrl**–**C** at this process group). If `-session` is specified, don't make the process a session leader. The default is `-session`.

This parameter is typically used for the shell:

```
[+session] esh
```

Bootfile

When building a bootable filesystem, you must specify a bootfile via the *`physical`* (p. 1253) or *`virtual`* (p. 1257) attribute. Note that the bootfile must be the first file specification within the buildfile. If the first character of the bootfile is a left square bracket (`[`), a list of configuration attributes is given in the same syntax as the buildfile. The list of attributes is terminated by a right square bracket (`]`). The allowed attributes, and their formats, are:

`attr=`*image_attribute*

Specify an attribute to add to the image. These attributes are processed after the -l ("el") command-line options and the buildfile, but you normally use the ? prefix on the *image_attribute*, so that it doesn't override anything explicitly set by the -l option or the buildfile.

Copyright © 2014, QNX Software Systems Limited

+|-big_pages

Align binaries and shared libraries at the appropriate address, based on the size of the UIP text and the sizes specified by the `pagesizes` attribute. You can specify these attributes in the buildfile or in the bootfile. For more information, see the *big_pages* (p. 1247) buildfile attribute.

default_image=*addr_space_spec*

Set the defaults for the *image* (p. 1249) file attribute (see above).

default_ram=*addr_space_spec*

Set the defaults for the *ram* (p. 1254) file attribute (see above).

filter=*image_filter_spec*

After the image has been created, run *image_filter_spec*. The following formatting codes are expanded:

- `%i` — the name of the output image file.
- `%I` (uppercase i) — the name of the input file (see below).
- `%s` — the offset of the startup header in the image file, in hex with a leading `0x`.
- `%a` — any arguments from the `physical` or `virtual` attribute.

See "*Image filter* (p. 1264)," below for an example of using the *image_filter_spec*.

len=*boot_length*

The *boot_length* parameter gives the amount of space to leave at the front of the image file (before the actual image filesystem) for system header information or boot prefix code. This is the minimum amount of space to reserve. If the boot prefix code following the bootfile attributes is larger than the number given here, the size of the boot prefix code is used instead. The default is zero.

notloaded=*length*

In some systems (such as IBM OpenBIOS), the system header information isn't loaded into memory and doesn't contribute to the memory offsets where things are placed (the base address of the image being set by the `image` attribute in the buildfile). This attribute specifies the size of the information that isn't going to be loaded into memory. The default is zero.

paddr_bias=*number*

Copyright © 2014, QNX Software Systems Limited

On some CPUs, the hardware reserves a range of virtual addresses that map one-to-one with physical address. This attribute lets `mkifs` know how to translate a virtual address to its physical location in memory via the formula:

```
phys_addr = virt_addr + number
```

The default is zero.

`pagesize=`*`size`*

Set the size of a page in the image filesystem. The `mkifs` utility aligns various structures to a multiple of this. The default is 4 KB.

`pagesizes=`*`size`*`[,`*`size`*`]...`

Define the page sizes for use with the `big_pages` attribute. You can specify these attributes in the buildfile or in the bootfile. For more information, see the *`pagesizes`* (p. 1252) buildfile attribute.

`vboot=`*`addr`*

When building a virtual system, the paging hardware is sometimes turned on by the startup code (e.g. x86 architecture), as opposed to `procnto`. In the first case, this option tells `mkifs` what base virtual address to use for the bootstrap executables. This option has no effect when building a physical system. The default is none.

Following the closing square bracket character (`]`) of the bootfile attributes, `mkifs` searches for the string `boot`. If it's found, `mkifs` considers all data immediately following, through to the end of the file, to be boot prefix code. This data is placed at the start of the image file. If the `len` attribute was specified and is larger than the size of the boot prefix code, the image file is padded out to the size given.

Image filter

You can specify an image filter within the specification for the bootfile, and optionally specify macro expansions to it. These macro expansions are documented above, in the *`filter`* (p. 1248) description.

The following image filters are currently available:

`mkifsf_elf`

Wrap the entire image in an ELF section.

`mkifsf_openbios`

Patch the header at the beginning of the image for IBM's OpenBIOS.

`mkifsf_srec`

Copyright © 2014, QNX Software Systems Limited

Convert the image into S-Record format. This filter accepts the following options:

- -b — generate only 4-byte address records.
- -c — omit the carriage-return character at the ends of lines.
- -l — omit the linefeed character at the ends of lines.

Generally, image filters are expected to take the file specified by the `%i` variable and modify it in place. If this isn't possible (e.g. the file changes size as a result of the filter program), specifying `%I` causes `mkifs` to store the original file in a temporary filename (named by `%I`), and expect the modified file in the filename given by `%i`. This happens only when the `%I` macro expansion is specified.

Linker specification

The linker specification lets you control how the linker command line is built when `mkifs` needs to turn a relocatable object into an executable running at a particular address. It operates much like a *printf()* format string, with characters being copied from the format to the command line until a percent character (%) is found. The following formatting codes after a percent character are supported:

%h

The address to place the executable header at, in hexadecimal.

%t

The address to place the text segment at, in hexadecimal. This is **%h** plus the amount of space for the executable header structures.

%d

The address to place the data segment at, in hexadecimal. This value may be zero, in which case the data is placed immediately following the text segment.

%o

The name of the output executable file, as a string.

%i

The name of the input relocatable file, as a string.

%(

Open a conditional section. Following the opening parenthesis, **(**, are: a single character indicating the variable that the section is conditional on; one of the usual conditional operators from the C language; a constant; and finally a comma.

Copyright © 2014, QNX Software Systems Limited

The contents of the variable are compared against the constant and if the result is true, the text following the comma is included in the command string being built. If the comparison is false, the contents of the string following the comma are omitted.

The conditional is terminated by percent character followed by a closing parenthesis, `%)`. You can nest conditionals. The variables that you can test are:

Variable:	Value:
e	0 == little endian 1 == big endian
d	Data segment address
f	0 == startup file 1 == bootstrap file 2 == normal file
h	Executable header address
m	Machine number from the ELF header
v	0 == file linked physically 1 == file linked virtually
V	0 == physical system 1 == virtual system

`%)`

Terminate a conditional section.

Here's the default linker command specification for `mkifs`:

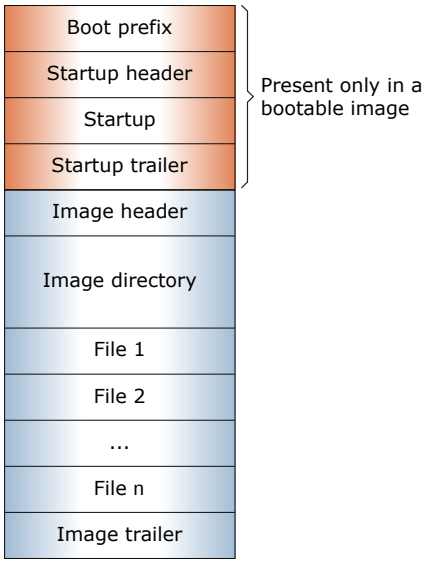
```
static char default_linker[] = {
        "qcc"
        " -bootstrap -nostdlib -Wl,--no-keep-memory -Vgcc_nto"
        "%(m==3,x86%)%(m==6,x86%)"
        "%(m==40,arm%)"
        "%(m!=3,%(m!=6,%(e==0, -EL%)%(e==1, -EB%)%)%)"
        "%(h!=0, -Wl,-Ttext -Wl,0x%t%)%(d!=0, -Wl,-Tdata -Wl,0x%d%)"
        " -o%o %i"
        "%[M -L%^i -Wl,-uinit_%n -lmod_%n%]"
};
```

For the meaning of the parameters specified, see *gcc* (p. 889).

Output image format

The image created by `mkifs` has the following layout:

Copyright © 2014, QNX Software Systems Limited



- The boot prefix is generated based on the bootfile that you specified with the `virtual=` or `physical=` attribute.
- The boot prefix, startup header, startup, and startup trailer are present only in a bootable image.
- A checksum for the startup information is stored in the startup trailer.
- A checksum for the image is stored in the image trailer.

Although it isn't necessary to have a detailed understanding of the format of an image to make one, a general understanding is worthwhile.

Boot prefix

The first section (called the *boot prefix*) is controlled by the bootfile that you specified in the `virtual=` or `physical=` attribute. For many systems this section doesn't occupy any space in the image. When it's present, it's typically used to address one of the following issues:

- The IPL code that transfers control to the image doesn't set up the processor state in a way that's compatible with startup. In this case, this section contains code that does that work. If you've written your own IPL, it ensures the processor is in a suitable state before jumping to startup, and this section is empty.

  A boot on a standard x86 PC is a good example of the need for placing code here. When a PC boots, it transfers control while in 16-bit real mode. The startup program assumes the processor is running in 32-bit protected mode. So, an image with a PC BIOS boot contains code here that switches the processor into 32-bit protected mode. It also does a series of BIOS calls to gather information from the BIOS, since the protected mode startup program is unable to make any BIOS calls itself.
- The image is wrapped or encapsulated within another data structure used by an IPL. To ensure proper alignment of executables on page boundaries, `mkifs` needs to know how large the wrapper is at the beginning of the image. In this case, `mkifs`

Copyright © 2014, QNX Software Systems Limited

creates a zero-filled region for the boot prefix, which an external program (the image filter) modifies as a post-processing pass over the image.

An example of this is a network boot in which the image needs to be wrapped in something that was loaded in its entirety into memory on the target (e.g. ELF object file structures). In this case, an external program makes a copy of the image, adding information to the front, and possibly the end, of the image. If the wrapper prefix is a small fixed size, you may wish to include a boot prefix that's zero-filled, which an external program can overwrite. This saves you having to make a file copy of a large image to append to the wrapper. You can always append a wrapper directly to the end of an image file.

Startup header

This section contains information about the image, which is used by our IPL and startup programs.

Part of this section is written to by `mkifs`. Another part is set to zero, and is written to by the IPL code to pass data (determined at runtime) to startup. The data is in the form of a set of structures (for more information, see Customizing IPL Programs in *Building Embedded Systems*).

If an image isn't bootable, this section is omitted.

Startup

This section contains the code and data for the startup program. This code must be executed in RAM. If the image is in ROM/FLASH, our standard IPL code uses information in the startup header to always copy the startup into RAM and transfer control to it there.

If an image isn't bootable, this section is omitted.

Startup trailer

A checksum for use by startup. If an image isn't bootable, this section is omitted.

Image header

Information on the image filesystem that follows.

Image directory

A series of directory entries for each file in the image filesystem.

Files

The files within the image filesystem. Executables that are executed in place are aligned on page boundaries. An attempt is made to fill any holes created by this alignment with small data files that have no alignment needs.

Image trailer

A checksum for the image.

Copyright © 2014, QNX Software Systems Limited

Notes on XIP versus copy

You can apply the `code=copy|uip` and `data=copy|uip` attributes to executables in the image.

Shared objects currently assume `code=uip` and `data=copy`. To get the effect of `code=copy`, manually copy the shared object to `/dev/shmem` and set ***LD_LIBRARY_PATH*** as appropriate. To save ROM/Flash space, compress the shared object in the image filesystem and decompress it into `/dev/shmem`.

When an executable is run, these attributes determine whether the code or data for that executable is to be used in place or copied to RAM. An image filesystem may exist in either RAM or linearly addressable ROM/FLASH. Images in RAM are usually loaded from a device that isn't linearly addressable. This includes disk boots, network boots, and boots from bank-switched ROM/FLASH devices. It also includes any image that's been compressed.

For example, a compressed image may reside in linearly addressable flash, but it can't be used until it's decompressed into RAM. The following combinations exist for any image in RAM:

Code	Data	Comments
`uip`	`uip`	Run once (default)
`uip`	`copy`	Run multiple
`copy`	`uip`	Run once but wasteful
`copy`	`copy`	Run multiple but wasteful

The default assumes that you want to run both code and data in place. This would be fine for an executable that has a large amount of static data and that you need to run only once (e.g. a resource manager or driver that starts when the system boots). Running data in place modifies the only copy of the data as the program runs, so you can't start the program again.

In this case, the file's sticky bit won't be set. This identifies the executable to the QNX Neutrino process manager, which will prevent the program from running more than once, thus avoiding the possibility of running a program with corrupted static data.

That's why you have to specify `data=copy` if you want to run the executable multiple times.

The two cases listed as “wasteful” fall out of the combinations but they provide no additional capabilities and waste memory by copying the code unnecessarily. Since the code is read-only and can't be modified, it can always be used in place.

Copyright © 2014, QNX Software Systems Limited

If you're creating an image in ROM/FLASH, the following combinations exist:

Code	Data	Comments
`uip`	`uip`	Run once
`uip`	`copy`	Run multiple (default)
`copy`	`uip`	Run once
`copy`	`copy`	Run multiple (slow ROM/FLASH)

You have to specify both *`image`* (p. 1249) and *`ram`* (p. 1254) file attributes if you want to create the image in ROM/FLASH; otherwise the process manager assumes that the image is in RAM. The `data=copy` attribute is assumed for an image in ROM/Flash.

The cases where code is copied may seem wasteful (as in the RAM image example) but it may make sense for systems where the ROM/FLASH is slow — perhaps it has an 8-bit interface or requires extra wait states for access to it. In that case, you may wish to copy it to RAM, so that it executes faster.

Examples:

Here's a very simple buildfile that specifies the operating system, a console driver, and a shell:

```
[virtual=x86,bios] .bootstrap = {
    startup-bios
    PATH=/proc/boot procnto
}

[+script] .script = {
    devc-con -n9 &
    reopen /dev/con1
    [+session] esh &
}
libc.so
[data=copy]
devc-con
esh
[type=link] /usr/lib/ldqnx.so.2=/proc/boot/libc.so
```

The runtime linker is expected to be found in a file called `ldqnx.so.2`, but the runtime linker is currently contained within the `libc.so` file, so we make a process-manager symbolic link to it.

You can now build an image from the above, like this (assuming that the buildfile is called `simple.bld`, and that we want the resultant image to be called `simple.ifs`):

```
mkifs simple.bld simple.ifs
```


Copyright © 2014, QNX Software Systems Limited

Here's a buildfile with EIDE disk support:

```
[virtual=x86,bios +compress] .bootstrap = {
    startup-bios
    PATH=/proc/boot procnto
}
[+script] .script = {
    devc-con -e &
    devb-eide &
    reopen /dev/con1
    [+session] PATH=/proc/boot esh &
}
libc.so
libcam.so
cam-disk.so
io-blk.so
fs-qnx4.so

[data=copy]
devc-con
esh
ls
devb-eide
[type=link] /usr/lib/ldqnx.so.2=/proc/boot/libc.so.3
```

The following example includes an inline `/etc/hosts` file that's used to resolve addresses used at boot time by programs such as `fs-nfs3`; it also shows how you can pass environment variables to different commands:

In a real buildfile, you can't use a backslash (\) to break a long line into shorter pieces, but we've done that here, just to make the buildfile easier to read.

```
[image=0x1f0000]
[virtual=armle-v7,raw] .bootstrap = {
    startup-my_board-smp -v -Nmy_board-5 -D0x800003f8^0.9600
    PATH=/proc/boot:/bin:/usr/bin:/sbin:/usr/sbin \
LD_LIBRARY_PATH=/proc/boot:/lib:/usr/lib:/lib/dll \
procnto-600-smp -v
}

[+script] startup-script = {
    # To save memory, make everyone use the libc in the boot
    # image! For speed (fewer symbolic lookups), we point to
    # libc.so.3 instead of libc.so.

    procmgr_symlink ../../proc/boot/libc.so.3 /usr/lib/ldqnx.so.2

    pci-raven &
    waitfor /dev/pci

    io-pkt-v4 -dtulip irq=2,media=9,vid=0x1011,did=0x9 -ptcpip &
    if_up -p en0
    ifconfig en0 my_board-5 up
    if_up en0

    fs-nfs3 -ru ra:/my_system /my_system &
    waitfor /my_system/target/qnx6/armle-v7/usr/sbin/slogger 360

    # setup environment variables
    TZ=est05edt04

    procmgr_symlink /my_system/target/qnx6/armle-v7/bin /bin
    procmgr_symlink /my_system/target/qnx6/armle-v7/lib /lib
    procmgr_symlink /my_system/target/qnx6/armle-v7/sbin /sbin
    procmgr_symlink /my_system/target/qnx6/armle-v7/usr/bin /usr/bin
    procmgr_symlink /my_system/target/qnx6/armle-v7/usr/sbin /usr/sbin
    procmgr_symlink /my_system/target/qnx6/armle-v7/usr/lib /usr/lib
    procmgr_symlink /my_system/target/qnx6/etc /etc

    slogger &
    waitfor /dev/slog
    devc-ser8250 -e -c1846200 -b 9600 0x800003f8,104 0x800002f8,103 &
    waitfor /dev/ser1
    pipe &
    waitfor /dev/pipe
    devc-pty &
    waitfor /dev/ptyp0
    mqueue &
    inetd &

    tinit
}

[type=link] /tmp = /dev/shmem
[type=link] /dev/con1 = /dev/ser1
```

Copyright © 2014, QNX Software Systems Limited

```
# Data files are created in the named directory
/etc/hosts = {
127.0.0.1       localhost
192.168.1.1     ra
192.168.1.111   my_board-5
}

# Include the current libc.so. It will be created as a real
# file using its internal SONAME, with libc.so being a
# symlink to it. The symlink will point to the last libc.so.*,
# so if an earlier libc is needed (e.g. libc.so.2), add it
# before libc.so.

libc.so.2
libc.so
devn-tulip.so
libsocket.so

[data=uip]
pci-raven
io-pkt-v4

[data=copy]
if_up
ifconfig
fs-nfs3
```

For more examples, see ${***QNX_TARGET***}/${***PROCESSOR***}/build.

Environment variables:

PROCESSOR

Specifies the target CPU. If not set, the default is the same as the CPU of the host system (e.g. x86).

MKIFS_PATH

Specifies a colon-separated list of directories to search for host files that to be included in the image. The default value consists of:

1. The current working directory, if the filename contains a slash (/) but doesn't start with a slash.
2. ${***QNX_TARGET***}/${***PROCESSOR***}/sbin
3. ${***QNX_TARGET***}/${***PROCESSOR***}/usr/sbin
4. ${***QNX_TARGET***}/${***PROCESSOR***}/boot/sys
5. ${***QNX_TARGET***}/${***PROCESSOR***}/bin
6. ${***QNX_TARGET***}/${***PROCESSOR***}/usr/bin
7. ${***QNX_TARGET***}/${***PROCESSOR***}/lib
8. ${***QNX_TARGET***}/${***PROCESSOR***}/lib/dll
9. ${***QNX_TARGET***}/${***PROCESSOR***}/usr/lib

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mkimage

Build a socket image from individual files (QNX)

Syntax:

```
mkimage [ -b blocksize ] -o outputfile file…
```

Runs on:

Linux, Microsoft Windows

Options:

-b *blocksize*

The blocksize used when padding files. The default is 64K.

-o *outputfile*

The name of the output file.

Description:

The `mkimage` utility builds a socket image from individual files. The command line is parsed, and the bootable image file(s) are placed first in the resultant output file, followed by embedded filesystem files, then any other files on the command line.

All files are padded up to the block size specified on the command line (padding is done with `0xFF`, the default erased state of flash). If no blocksize is specified, the default is 64K.

Copyright © 2014, QNX Software Systems Limited

mkqnx6fs

Format a Power-Safe filesystem (QNX Neutrino)

You must be logged in as `root` to run this utility.

Syntax:

```
mkqnx6fs [-BEq] [-b blocksize] [-e endian] [-g groups]
         [-i inodes] [-n blocks] [-O options] [-o options]
         [-r percent] [-T type] [-u uuid] [-v vol_name] host
```

Runs on:

QNX Neutrino

Options:

-B

Rewrite only the boot loader; don't touch anything in the filesystem (in particular don't reformat it). You would use this to upgrade to new boot loader code.

-b *blocksize*

Set the logical blocksize of the filesystem. You can specify the size in bytes or in kilobytes, as follows:

Bytes	Kilobytes
512	—
1024	1k
2048	2k
4096	4k

The default is 1024; the case of the “k” is ignored. Varying the blocksize can control various types of fragmentation as well as determine the maximum file size supported.

-E

Copyright © 2014, QNX Software Systems Limited

Enable support for file data encryption. In order to use encryption, you must specify crypto=enable for *fs-qnx6.so* (p. 823) and then use *fsencrypt* (p. 834) to manage the encryption.

-e ***endian***

Set the endian layout of the on-disk filesystem. Valid values are `big` or `little`; by default the filesystem uses the native endian-ness of the CPU.

-g ***groups***

Set the number of allocation groups to subdivide the filesystem; by default a value (from 1 to 16) is selected based on filesystem size. An allocation group is a logical concept, not a physical segregation.

-i ***inodes***

Set the maximum number of inodes in the filesystem. Each unique file or directory requires an inode.

-n ***blocks***

Set the number of logical blocks in the filesystem. This is the total size of the filesystem, from which space is first allocated to the system bitmap and inodes files (so the number of user-accessible blocks will be slightly less than the specified value).

By default, `mkqnx6fs` makes the filesystem fully occupy the specified host (e.g. it determines the number of blocks from the size of the disk partition).

-O ***options***

("Oh") Set(+) or unset(-) boot options:

- quiet — stop the boot loader from doing any output, disable the boot image selection menu, and silently boot the default image.
- cls — clear the screen first (in case the BIOS didn't do it earlier, and there isn't enough room for the menu).

The default is -O-quiet,-cls.

You can use this option with -B to just update the loader and options.

-o ***options***

Set(+) or unset(-) filesystem options:

- lfncksum — enable a `cksum` algorithm on long filenames (longer than 27 characters), which greatly improves their lookup performance.

The default is -o+lfncksum.

Copyright © 2014, QNX Software Systems Limited

This default is incompatible with the 6.4.0 version of the Power-Safe filesystem. If you wish to format a filesystem that can be mounted read-write by 6.4.0, you must specify -o-lfncksum; otherwise it will allow only read-only mounting.

-q

Operate quietly; don't prompt for confirmation and don't display the resulting configuration of the new filesystem. Without this option, `mkqnx6fs` will confirm that you meant to format if the host is a block-special device or is currently mounted.

-r ***percent***

Set the percentage of the filesystem to reserve to prevent it from becoming completely full. In general, filesystem performace degrades when the disk is nearly full; this option just makes ENOSPC happen prematurely to stop this. The default is 3%.

-T ***type***

Set the expected usage type of the filesystem; valid values are `desktop`, `runtime`, and `media`. This type is used to pick the appropriate blocksize, number of allocation groups, and number of inodes. It's a hint that's intended to replace explicit -b, -g, -i, and -r values.

-u ***uuid***

Specify a 128-bit UUID for the filesystem, in the UUID "8-4-4-4-12" format. If you don't specify a UUID, `mkqnx6fs` generates a random, time-based (version 4 UUID) value.

-v ***vol_name***

Specify a volume name of up to 16 characters.

You can't specify both the -u and -v options.

host

The host of the new filesystem. You can specify this as a block-special device or partition (e.g. `/dev/hd0t76`), as a regular file, or as the root directory of a mounted `fs-qnx6` filesystem (which will be resolved to the real host device).

Copyright © 2014, QNX Software Systems Limited

Description:

The `mkqnx6fs` utility creates a fresh `fs-qnx6` filesystem on the specified host (typically a hard disk partition, although you can create an image inside a regular file).

The integer fields of the filesystem are maintained as either all little-endian or all big-endian, as dictated by the -e option. Thus no CPU architecture pays a byte-swapping penalty for local disks. The filesystem detects the endian-ness and swaps if necessary, so you can move a disk across platforms (with a slight penalty in performance).

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Examples:

```
# mkqnx6fs /dev/hd0t76
All files on /dev/hd0t76 will be lost!
```

Copyright © 2014, QNX Software Systems Limited

```
Confirm filesystem re-format (y) or (n): y
Format fs-qnx6: 8040524 blocks, 62816 inodes, 8 groups
```

Exit status:

0

The filesystem was formatted successfully.

1

An error occurred (a descriptive message is written to *stderr*).

Copyright © 2014, QNX Software Systems Limited

mkqnx6fsimg

Build a Power-Safe filesystem image (QNX)

Syntax:

```
mkqnx6fsimg [option...] [buildfile] [directory] [outputfile]
mkxfs -t qnx6fsimg [option...] [buildfile] [directory]
[outputfile]
```

Runs on:

Linux, Microsoft Windows

Options:

-D

Treat undeclared intermediate directories as errors. If there's a target filesystem entry of `/x/y`, and `/x` has never occurred explicitly in the buildfile, the patch file, the input directory, or as a child of a recursively included directory, then `/x` is considered an undeclared intermediate directory.

-d

Display warnings for undeclared intermediate directories.

-l *inputline*

("el") Process *inputline* before interpreting the buildfile. Input lines given to `mkqnx6fsimg` should be quoted to prevent interpretation by the shell. Multiple -l options are processed in the order specified. This option is especially useful for setting global attributes when the input is a directory only.

-n[n..]

Don't use timestamps in the files. Using the -n option permits identical images in binary format. One n strips timestamps from files that vary from run to run. More than one strips ALL time information, which is necessary on Windows NTFS with daylight saving time.

-p *patchfile*

Apply patching instructions from this file (see "*Patch files* (p. 1283)," below).

-r *root*

Copyright © 2014, QNX Software Systems Limited

Search the default paths in this directory before the default.

-v[v..]

Operate verbosely. Specifying additional v options increases the verbosity. The default is quiet operation.

Description:

The `mkqnx6fsimg` utility reads a text buildfile and/or a specified directory and produces a binary image file containing a Power-Safe (*`fs-qnx6.so`* (p. 823)) filesystem created from the given input. You can copy this file to target media at a later stage.

Don't confuse this command with *`mkqnx6fs`* (p. 1274), which initializes an empty Power-Safe filesystem and is available for Neutrino hosts only.

You specify the input and output on the command line:

buildfile

The filename of the buildfile that describes the properties and contents of the Power-Safe filesystem; use a hyphen (`-`) to specify standard input (the default).

directory

The root of a directory hierarchy to be appended to the file list specified in *buildfile* (if any). The default is no directory.

outputfile

The filename of the image file containing the Power-Safe filesystem; use a hyphen (`-`) to specify standard output (the default). Note that you can specify the output file only if you specified at least either a buildfile or a directory.

If you don't specify either a buildfile or a directory, a buildfile is expected as input from standard input. The output is always an image file; if you don't specify *outputfile*, image-file data will be produced on standard output.

A Power-Safe filesystem *must* always be exactly the same size as the partition (or unpartitioned medium) it resides on. If it differs in size, then it will be reported as being corrupted. Be also aware that many [pseudo] hard-disk drives report a virtual geometry of 255 heads and 63 sectors per track, providing the maximum of 16065 sectors per cylinder. Since partitions are always defined in units of entire cylinders, any Power-Safe filesystem image intended to be put onto a [pseudo] hard-disk should have a sector count that's an integer

Copyright © 2014, QNX Software Systems Limited

multiple of the reported number of sectors per cylinder. This requirement is relaxed for resizable images; see the *alloc_bnd=size_spec* (p. 1284) attribute.

Buildfiles

The `mkqnx6fsimg` command uses the same buildfile grammar as `mkifs`, but supports a different set of attributes. The buildfile is basically just a list of files that you want to be included in the Power-Safe image file when it's built by `mkqnx6fsimg`. As well as identifying the files to be included, you can specify various attributes that are used to set parameters of the filesystem and the files in it. For example, you can specify the maximum size of the filesystem, or the user and group IDs of the individual files.

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

  ```
  [attribute] A
  B
  C
  ```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

  ```
  [attribute]
  A
  B
  C
  ```

Attributes provide information about the file following the attribute. They are enclosed in square brackets; when combining attributes (e.g., to specify both the user ID and the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
```

Copyright © 2014, QNX Software Systems Limited

```
[uid=5 gid=5] filename
# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

Boolean attributes

Those prefixed with a plus (+) or minus (-) sign.

Value attributes

Those ending with an equals sign (=) followed by a value. Don't put any spaces around the equals sign.

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example:

```
?+bigendian
```

sets the `+bigendian` attribute only if `+bigendian` or `-bigendian` hasn't already been set.

The *file_specification* takes one of the following forms:

path

The file is copied from the host to the location in the image defined by the *prefix* (p. 1288) attribute. If *path* isn't absolute, `mkqnx6fsimg` looks for it in the locations identified by the *search* (p. 1289) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces (`{ }`)—the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mkqnx6fsimg` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mkqnx6fsimg` doesn't interpret a pound sign (#) in an inline file as the beginning of a comment. The syntax of the inline file depends on what it's used for on the target system.

Closing braces (`}`) and backslashes (`\`) in an inline file must be escaped with a backslash.

Copyright © 2014, QNX Software Systems Limited

You can enclose a filename in double quotes (") if it includes spaces or unusual characters.

Patch files

Patch files let you override the user ID, group ID, and permissions of certain files, depending on their location and filename pattern. Patches are applied after all files have been collected (from the buildfile and/or the specified directory). Consequently, patch files can override settings specified in the buildfile.

Patch files must contain only lines of the form:

```
#comment
```

or:

```
type:path:pattern:uid:gid:perms
```

In comment lines, # must be the very first character. The entire line is regarded as a comment and is ignored.

The *type* is either `d` or `f`, optionally followed by `r`. Type `d` patches are applied only to directories, and type `f` patches are applied only to files. An `r` indicates that the patch should be applied recursively within *path*; without `r`, the patch is applied to *path* only.

The *pattern* is a filename pattern that specifies which files to apply the patch to. The *uid* and *gid* must be decimal numbers, while *perms* must be an octal number (see *chmod* (p. 124)). Note that it isn't possible to set only the user ID, group ID, or permissions; for each match, all three are affected.

Attributes

In `mkqnx6fsimg` buildfiles, the following attributes are supported:

- *alloc_bnd=size_spec* (p. 1284)
- *+|-bigendian* (p. 1284)
- *blksize=size_spec* (p. 1285)
- *+|-boot_cls* (p. 1285)
- *+|-boot_quiet* (p. 1285)
- *cd=path* (p. 1285)
- *dperms=perm_spec* (p. 1285)
- *filter=filter_spec* (p. 1285)
- *+|-followlink* (p. 1286)
- *+|-fsys_lfncks* (p. 1286)
- *gid=id_spec* (p. 1286)
- *mtime=time_spec* (p. 1286)
- *num_blocks=size_spec* (p. 1287)
- *num_groups=size_spec* (p. 1287)

Copyright © 2014, QNX Software Systems Limited

- *num_inodes=size_spec* (p. 1287)
- *num_sectors=size_spec* (p. 1287)
- *+|-optional* (p. 1288)
- *perms=perm_spec* (p. 1288)
- *prefix=path* (p. 1288)
- *reserve=number* (p. 1288)
- *search=path[:path...]* (p. 1289)
- *sector_size=size_spec* (p. 1289)
- *type=file_type* (p. 1289)
- *uid=id_spec* (p. 1290)
- *usage=usage* (p. 1290)
- *uuid=uuid* (p. 1290)
- *vol_name=string* (p. 1290)

The following attributes are recognized, but not semantically supported by `mkqnx6fsimg`:

- `mountperms=`*perm_spec*

Although default values are defined to generate a 256 MB image, you should explicitly specify at least the image size, preferably by specifying at least `num_sectors`, to ensure that the image produced is fully compatible with your specific target device.

An OR-bar indicates that either the first element or the second element must be present, but not both (e.g., `+|-bigendian` means either `+bigendian` or `-bigendian`, but not `+-bigendian`).

`alloc_bnd` attribute

`alloc_bnd=`*size_spec*

Specify an allocation boundary for the Power-Safe filesystem, indicating that a resizable image is to be generated. When you use the `alloc_bnd` attribute, `mkqnx6fsimg` creates a filesystem image as would be output by *dprepresize* (p. 641). This image *isn't* a valid Power-Safe filesystem, but can later be resized with the *dresize* (p. 643) utility to fit potentially smaller media than specified at creation time. This attribute defines the *allocation boundary*; it ensures that in the generated image, no more than *size_spec* blocks will be used. The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). The default value is 0, which means "create a regular filesystem image."

`bigendian` attribute (boolean)

`+|-bigendian`

Set the byte order for the Power-Safe filesystem to either big (via `+bigendian`) or little (via `-bigendian`) endian. The default is host's native endian-ness.

Copyright © 2014, QNX Software Systems Limited

blksize attribute

```
blksize=size_spec
```

Set the block size for the Power-Safe filesystem. The block size determines the minimum allocatable amount in the filesystem and must be an integer multiple of the sector size. Valid values for the block size are 512, 1024, 2048, and 4096. *size_spec* is an integer, optionally followed by K, M, or G (the case doesn't matter). The default value is 1024.

boot_cls attribute (boolean)

```
+|-boot_cls
```

Clear (+) or don't clear (-) the screen while booting. The default is not to clear.

boot_quiet attribute (boolean)

```
+|-boot_quiet
```

If set, instructs the boot loader to silently boot the latest boot image, without presenting a boot-image selection menu. The default is not to boot quietly.

cd attribute

```
cd=path
```

Set the current working directory to the specified pathname before attempting to open the host file. The default is the directory from which you invoked mkqnx6fsimg.

dperms attribute

```
dperms=perm_spec
```

Set the access permissions of the directory. If specified as a number, the permissions are set to that number (just like the *chmod* (p. 124) command). If you specify the permissions as an asterisk (*), the host directory's permissions are used; for an inline directory, the permissions are 0755 (rwxr-xr-x). Otherwise, a symbolic mode string (which is a subset of chmod's) is used to delete, add, or set permissions. The symbolic mode string consists of:

1. a combination of u, g, o, and a
2. one of -, =, or +
3. a combination of r, w, x, s, g, and t.

You can include more than one symbolic mode string, separating them with a comma (,). The default dperms_spec is *.

filter attribute

```
filter=filter_spec
```

Copyright © 2014, QNX Software Systems Limited

Run the host file through the filter program specified, presenting the host file data as standard input to the program and using the standard output from the program as the data to be placed into the Power-Safe filesystem. Default is no filter. You can specify a `filter_spec` of none. This is useful if you need to override a global filter specification.

`followlink` attribute (boolean)

```
[+|-followlink]target_path=host_path
```

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mkqnx6fsimg` resolves the symbolic link and includes its target file or directory. If you specify `-followlink`, `mkqnx6fsimg` includes the symbolic link itself in the image. It's up to you to include in the image whatever the link points to.

`fsys_lfncks` attribute (boolean)

```
+|-fsys_lfncks
```

This attribute determines whether checksums are calculated for long filenames in the Power-Safe filesystem. The default is true.

`gid` attribute

```
gid=id_spec
```

Set the group ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the group ID is taken from the host file; for an inline file, the group ID is the group of the user running `mkqnx6fsimg`. The default value for this attribute is `*`.

`mountperms` attribute

This attribute is supposed to set the access permissions for the filesystem mountpoint. Since the Power-Safe filesystem has no provision to store mount information, this attribute is silently disregarded.

`mtime` attribute

```
mtime=time_spec
```

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (`*`), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```


Copyright © 2014, QNX Software Systems Limited

You must provide all six elements. The time is always interpreted as UTC.

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

`num_blocks` attribute

`num_blocks=`*`size_spec`*

Set the number of blocks in the Power-Safe filesystem. If you don't specify the number of blocks, then it's calculated from the number of sectors. If you have both the sector and the blocks count specified explicitly in the buildfile, be aware that the maximum number of blocks that a Power-Safe filesystem can have is:

((*num_sectors* * *sector_size*) - 16 KB) / *block_size*

The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). The default is 262128 (255K), which is the block count resulting from the above formula for the default number of sectors, using the default sector size and block size.

`num_groups` attribute

`num_groups=`*`size_spec`*

Set the number of allocation groups in the Power-Safe filesystem. You should let `mkqnx6fsimg` calculate a reasonable value for this attribute. The default is 4; the maximum is 64.

`num_inodes` attribute

`num_inodes=`*`size_spec`*

Set the number of file metadata entries in the filesystem. One individual inode is required for each file, directory, hard link, or symbolic link. This means that the inode count ultimately limits the number of files in the filesystem. The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). The default value is 32768.

`num_sectors` attribute

`num_sectors=`*`size_spec`*

Set the total number of sectors occupied by your image. This is the best way to ensure your image will fit a specific target device. The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). The default value is 524288 (512k); at the default sector size of 512 bytes, this equals an image size of 256 MB.

A Power-Safe filesystem *must* always be exactly the same size as the partition (or unpartitioned medium) it resides on. If it differs in size, then it will be reported as being corrupted. Be also aware that many [pseudo] hard-disk drives report a virtual geometry of 255 heads and 63 sectors per track, providing the maximum of 16065 sectors per cylinder. Since partitions are always defined

Copyright © 2014, QNX Software Systems Limited

in units of entire cylinders, any Power-Safe filesystem image intended to be put onto a [pseudo] hard-disk should have a sector count that's an integer multiple of the reported number of sectors per cylinder. This requirement is relaxed for resizable images; see the *alloc_bnd=size_spec* (p. 1284) attribute.

---

optional attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the embedded filesystem. If false, and the host file can't be found, output an error message and exit `mkqnx6fsimg`. The default is true.

perms attribute

```
perms=perm_spec
```

Set the access permissions of the file, hard link, or symbolic link. If specified as a number, the permissions are set to that number (just like the *chmod* (p. 124) command). If specified as an asterisk (`*`), the host file's permissions are used; for an inline file, permissions of 0666 are used. Otherwise, a symbolic mode string (which is a subset of `chmod`'s) is used to delete, add, or set permissions. The symbolic mode string consists of:

1. a combination of `u`, `g`, `o`, and `a`
2. one of `-`, `=`, or `+`
3. a combination of `r`, `w`, `x`, `s`, `g`, and `t`.

You can include more than one symbolic mode string, separating them with a comma (`,`). The default *perms_spec* is `*`.

When running on a Windows host, `mkqnx6fsimg` can't get the execute (`x`), setuid (“set user ID”), or setgid (“set group ID”) permissions from the file. Use the `perms` attribute to specify these permissions explicitly. You might also have to use the `uid` and `gid` attributes to set the ownership correctly. To determine whether or not a utility needs to have the setuid or setgid permission set, see its entry in the *Utilities Reference*.

---

prefix attribute

```
prefix=path
```

Set the prefix on the target file names. The default is the empty string.

reserve attribute

```
reserve=number
```


Copyright © 2014, QNX Software Systems Limited

Set the percentage of filesystem blocks to reserve for reclaim, as an integer in the range [0, 99]. The default value is 3.

`search` attribute

`search=`*`path`*`[:`*`path...`*`]`

This attribute specifies that `mkqnx6fsimg` should search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the Power-Safe filesystem. Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

`sector_size` attribute

`sector_size=`*`size_spec`*

Specify the sector size of the filesystem's target device. The *size_spec* is an integer, optionally followed by `K`, `M`, or `G` (the case doesn't matter). Valid values are 512 and 4096; the default is 512.

`type` attribute

`type=`*`file_type`*

Sets the type of the files being created in the Power-Safe filesystem. Allowable types are:

- `hlink`—a hard link (see below)
- `link`—a symbolic link
- `file`—a regular, everyday file (the default)
- `dir`—a directory

Specifying `[type=dir]` tells `mkqnx6fsimg` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```

You must always specify hard links using commands of the form:

```
[type=hlink] link-path=target-path
```

Copyright © 2014, QNX Software Systems Limited

where *link-path* is the new link to be created and *target-path* refers to an existing path in the Power-Safe filesystem that's being created. Note that it isn't possible to specify attributes such as `uid`, `gid`, or `permissions` for hard links. If you define any of these attributes together with the `[type=hlink]` attribute, they're silently ignored.

`uid` attribute

`uid=`*`id_spec`*

Set the user ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the user ID is taken from the host file; for an inline file, the user ID is the user running `mkqnx6fsimg`. The default value for this attribute is `*`.

`usage` attribute

`usage=`*`usage`*

Give a hint about the expected predominant usage of the Power-Safe filesystem. This hint will affect default values chosen for *blksize*, *num_blocks*, *num_groups*, and *num_inodes*. Valid values are:

- `d` or `desktop`
- `m` or `media`
- `r` or `runtime`

There is no default.

`uuid` attribute

`uuid=`*`uuid`*

Use this attribute to provide an explicit universally-unique identifier (UUID) for the Power-Safe filesystem. The UUID string must be in the form *XXXXXXXX-XXXX-XXXX-XXXXXXXX*, where *X* is a hexadecimal digit. The default is to use an IPv4/random-number-based UUID. Note that when you set the UUID, you must not provide a volume name.

`vol_name` attribute

`vol_name=`*`string`*

Set the volume name to the given string. The maximum volume label length is 16 characters. Note that when you specify a volume name, you must not provide a UUID. The default is to use the UUID.

Examples:

Here's a sample buildfile, `my_qnx6fs.bld`:

```
# A sample buildfile for mkqnx6fsimg
```


Copyright © 2014, QNX Software Systems Limited

```
[num_sectors=512k]
/home/thaupt
```

In this example, we've specified a sector count of 512*1024, which relates to an image size of 128 MB. The files and subdirectories from the `/home/thaupt` directory on the host system are to be recursively copied into the root directory of the Power-Safe filesystem. To create a Power-Safe filesystem image file using above buildfile, invoke `mkqnx6fsimg` as follows:

```
mkqnx6fsimg my_qnx6fs.bld my_qnx6fs.img
```

This creates the `my_qnx6fs.img` file containing the Power-Safe filesystem, which you could then copy to a target system's hard-disk partition as follows:

```
dd if=my_qnx6fs.img of=/dev/hd0t179 count=524288
```

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mkrcfs

Format a read-only compressed filesystem (QNX Neutrino)

You must be logged in as `root` to run this utility.

Syntax:

```
mkrcfs [options]... [directory] file
```

Runs on:

QNX Neutrino, Linux

Options:

-b 0..4

Specify the blocksize:

Value	Blocksize
0	4 KB
1	8 KB
2	16 KB (the default)
3	32 KB
4	64 KB

-c 1..9

The compression factor; the higher the factor, the slower the compression. The default is 9.

-D *type*

Dump filesystem structures (s for superblock, i for inodes).

-L *str*

A plain-text string (up to 32 bytes) to record in the superblock.

-l

("el") List the contents of *file* only. The directory isn't touched.

Copyright © 2014, QNX Software Systems Limited

-p *patch_file*

Apply patching instructions from this file (see "*Patch files* (p. 1293)," below).

-t none | lzo | ucl

The compression type to use:

Type	Compression	Decompression	Amount
lzo	Fast	Very fast	30–50% on executables
ucl	Slow	Fast	40–65% on executables

-v

Be verbose. You can specify up to -vvv to increase verbosity.

-x

Extract from *file* and create *directory*.

directory

Make a compressed filesystem of all files in this directory.

file

The output file representing the read-only compressed filesystem.

Description:

The `mkrcfs` utility creates a fresh read-only compressed filesystem (RCFS) on the specified host (typically a hard disk partition, although you can create an image inside a regular file).

Patch files

Patch files let you override the user ID, group ID, and permissions of certain files, depending on their location and filename pattern. Patches are applied after all files have been collected from the specified directory. Consequently, patch files can override the files' original settings.

Patch files must contain only lines of the form:

```
#comment
```

or:

```
type:path:pattern:uid:gid:perms
```

Copyright © 2014, QNX Software Systems Limited

In comment lines, # must be the very first character. The entire line is regarded as a comment and is ignored.

The *type* is either `d` or `f`, optionally followed by `r`. Type `d` patches are applied only to directories, and type `f` patches are applied only to files. An `r` indicates that the patch should be applied recursively within *path*; without `r`, the patch is applied to *path* only.

The *pattern* is a filename pattern that specifies which files to apply the patch to. The *uid* and *gid* must be decimal numbers, while *perms* must be an octal number (see *chmod* (p. 124)). Note that it isn't possible to set only the user ID, group ID, or permissions; for each match, all three are affected.

Summary of filesystem commands

The following table shows the shared objects and related commands for the filesystems:

Partition type	Filesystem	Shared object	Initialize with:	Check with:
1, 4, or 6	DOS	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
7	Windows NT[a]	*fs-nt.so* (p. 818)	N/A	N/A
11, 12, or 14	FAT32	*fs-dos.so* (p. 795)	*mkdosfs* (p. 1205)	*chkdosfs* (p. 111)
77, 78, or 79	QNX 4	*fs-qnx4.so* (p. 820)	*dinit* (p. 626)	*chkfsys* (p. 114)
131	Linux (Ext2)	*fs-ext2.so* (p. 807)	N/A	N/A
175	Apple Macintosh HFS or HFS Plus[a]	*fs-mac.so* (p. 809)	N/A	N/A
177, 178, or 179	Power-Safe	*fs-qnx6.so* (p. 823)	*mkqnx6fs* (p. 1274)	*chkqnx6fs* (p. 121)[b]
	Read-only compressed (RCFS)	*fs-rcfs.so* (p. 827)	*mkrcfs* (p. 1292)	N/A

[a] Read-only.

[b] Not usually necessary.

For more information, see the Filesystems chapter of the *System Architecture* guide.

Exit status:

Copyright © 2014, QNX Software Systems Limited

0

The filesystem was formatted successfully.

1

An error occurred (a descriptive message is written to *stderr*).

Copyright © 2014, QNX Software Systems Limited

mkrcfsimg

Build a read-only compressed filesystem (RCFS) image (QNX)

Syntax:

```
mkrcfsimg [option...] [buildfile] [directory] [outputfile]
mkxfs -t rcfs [option...] [buildfile] [directory] [outputfile]
```

Runs on:

Linux, Microsoft Windows

Options:

-D

Treat undeclared intermediate directories as errors. If there's a target filesystem entry of `/x/y`, and `/x` has never occurred explicitly in the buildfile, the patch file, the input directory, or as a child of a recursively included directory, then `/x` is considered an undeclared intermediate directory.

-d

Display warnings for undeclared intermediate directories.

-l *inputline*

("el") Process *inputline* before interpreting the buildfile. Input lines given to `mkrcfsimg` should be quoted to prevent interpretation by the shell. Multiple -l options are processed in the order specified. This option is especially useful for setting global attributes when the input is a directory only.

-n[n..]

Don't use timestamps in the files. Using the -n option permits identical images in binary format. One n strips timestamps from files that vary from run to run. More than one strips ALL time information, which is necessary on Windows NTFS with daylight saving time.

-p *patchfile*

Apply patching instructions from this file (see "*Patch files* (p. 1299)," below).

-r *root*

Copyright © 2014, QNX Software Systems Limited

Search the default paths in this directory before the default.

-v[v..]

Operate verbosely. Specifying additional v options increases the verbosity. The default is quiet operation.

Description:

The `mkrcfsimg` utility reads a text buildfile and/or a specified directory and produces a binary image file containing a read-only compressed filesystem (*`fs-rcfs.so`* (p. 827)) created from the given input. You can copy this file to target media at a later stage.

Don't confuse this command with *`mkrcfs`* (p. 1292), which formats an RCFS, but doesn't process a buildfile.

You specify the input and output on the command line:

buildfile

The filename of the buildfile that describes the contents of the RCFS; use a hyphen (`-`) to specify standard input (the default).

directory

The root of a directory hierarchy to be appended to the file list specified in *buildfile* (if any). The default is no directory.

outputfile

The filename of the image file containing the RCFS; use a hyphen (`-`) to specify standard output (the default). Note that you can specify the output file only if you specified at least either a buildfile or a directory.

If you don't specify either a buildfile or a directory, a buildfile is expected as input from standard input. The output is always an image file; if you don't specify *outputfile*, image-file data will be produced on standard output.

Buildfiles

The `mkrcfsimg` command uses the same buildfile grammar as `mkifs`, but supports a different set of attributes. The buildfile is basically just a list of files that you want to be included in the RCFS image file when it's built by `mkrcfsimg`. As well as identifying the files to be included, you can specify various attributes that are used to set parameters of the filesystem and the files in it. For example, you can specify the filesystem's label, or the compression algorithm for individual files.

Copyright © 2014, QNX Software Systems Limited

You can't use a backslash (\) to break long lines into smaller pieces.

In a buildfile, a pound sign (#) indicates a comment; anything between it and the end of the line is ignored. There must be a space between a buildfile command and the pound sign.

Each line is in the form:

```
[attributes] file_specification
```

where the attributes (with the enclosing square brackets) and the file specification are both optional.

You can use an attribute:

- on the same line as a filename, in which case the attribute modifies only that file. In this example, the attribute modifies only file A:

  ```
  [attribute] A
  B
  C
  ```

- on a line by itself, in which case the attribute modifies all subsequent files. In this example, the attribute modifies files A, B, and C:

  ```
  [attribute]
  A
  B
  C
  ```

Attributes provide information about the file following the attribute. They are enclosed in square brackets; when combining attributes (e.g., to specify both the user ID and the group ID), enclose both attribute tokens in the same pair of square brackets. For example:

```
# correct way
[uid=5 gid=5] filename
# incorrect way
[uid=5] [gid=5] filename
```

There are two types of attributes:

Boolean attributes

Those prefixed with a plus (+) or minus (-) sign.

Value attributes

Those ending with an equals sign (=) followed by a value. Don't put any spaces around the equals sign.

Copyright © 2014, QNX Software Systems Limited

A question mark (?) before an attribute makes the setting conditional. The attribute is set only if it hasn't already been set. For example:

```
?+followlink
```

sets the `+followlink` attribute only if `+followlink` or `-followlink` hasn't already been set.

The *file_specification* takes one of the following forms:

path

The file is copied from the host to the location in the image defined by the *prefix* (p. 1303) attribute. If *path* isn't absolute, `mkrcfsimg` looks for it in the locations identified by the *search* (p. 1303) attribute.

target_path=host_path

The specified file or contents of the specified directory are fetched from the host filesystem and placed into the image.

target_path={contents}

An *inline* definition. The contents of the file are listed within the buildfile itself, enclosed in braces (`{ }`)—the file doesn't exist on the host system anywhere. The contents of the inline file can't be on the same line as the opening or closing brace.

The `mkrcfsimg` utility doesn't parse the contents of an inline file for anything but the closing brace. For example, `mkrcfsimg` doesn't interpret a pound sign (#) in an inline file as the beginning of a comment. The syntax of the inline file depends on what it's used for on the target system.

Closing braces (`}`) and backslashes (`\`) in an inline file must be escaped with a backslash.

You can enclose a filename in double quotes (`"`) if it includes spaces or unusual characters.

Patch files

Patch files let you override the user ID, group ID, and permissions of certain files, depending on their location and filename pattern. Patches are applied after all files have been collected (from the buildfile and/or the specified directory). Consequently, patch files can override settings specified in the buildfile.

Copyright © 2014, QNX Software Systems Limited

Patch files must contain only lines of the form:

```
#comment
```

or:

```
type:path:pattern:uid:gid:perms
```

In comment lines, # must be the very first character. The entire line is regarded as a comment and is ignored.

The *type* is either `d` or `f`, optionally followed by `r`. Type `d` patches are applied only to directories, and type `f` patches are applied only to files. An `r` indicates that the patch should be applied recursively within *path*; without `r`, the patch is applied to *path* only.

The *pattern* is a filename pattern that specifies which files to apply the patch to. The *uid* and *gid* must be decimal numbers, while *perms* must be an octal number (see *chmod* (p. 124)). Note that it isn't possible to set only the user ID, group ID, or permissions; for each match, all three are affected. As mentioned above, the RCFS ignores any settings for the user ID and group ID.

Attributes

In `mkrcfsimg` buildfiles, the following attributes are supported:

- *block_size=bsize* (p. 1301)
- *cd=path* (p. 1301)
- *+|-compress* (p. 1301)
- *compress_type=compression_type* (p. 1301)
- *dperms=perm_spec* (p. 1301)
- *filter=filter_spec* (p. 1301)
- *+|-followlink* (p. 1302)
- *gid=id_spec* (p. 1302)
- *label=string* (p. 1302)
- *mtime=time_spec* (p. 1302)
- *+|-optional* (p. 1302)
- *perms=perm_spec* (p. 1303)
- *prefix=path* (p. 1303)
- *search=path[:path...]* (p. 1303)
- *type=file_type* (p. 1303)
- *uid=id_spec* (p. 1304)
- *verifier=verifier_type* (p. 1304)
- *zone=number* (p. 1304)

An OR-bar indicates that either the first element or the second element must be present, but not both (e.g., `+|-followlink` means either `+followling` or `-followlink`, but not `+-followlink`).

Copyright © 2014, QNX Software Systems Limited

block_size attribute

```
block_size=bsize
```

Set the block size for the RCFS; *bsize* can be 0 to use the default, or the size in kilobytes: 4, 8, 16 (the default), 32, or 64.

cd attribute

```
cd=path
```

Set the current working directory to the specified pathname before attempting to open the host file. The default is the directory from which you invoked `mkrcfsimg`.

compress attribute (boolean)

```
+|-compress
```

Enable or disable compression, using LZO if enabled.

compress_type attribute

```
compress_type=compression_type
```

Set the compression type:

Value	Type	Compression	Decompression	Amount
0	None	Very fast	Very fast	0%
1, 2	LZO	Fast	Very fast	30–50% on executables
3	UCL	Slow	Fast	40–65% on executables

Specifying a compression type of 0 is the same as specifying the `-compress` attribute. Compression types 1 and 2 are identical.

dperms attribute

```
dperms=perm_spec
```

Set the access permissions of the directory. See the *perms* (p. 1303) attribute for more information.

filter attribute

```
filter=filter_spec
```

Run the host file through the filter program specified, presenting the host file data as standard input to the program and using the standard output from the program as the data to be placed into the RCFS. Default is no filter. You can specify a `filter_spec` of none. This is useful if you need to override a global filter specification.

Copyright © 2014, QNX Software Systems Limited

`followlink` attribute (boolean)

```
[+|-followlink]target_path=host_path
```

If you specify `+followlink` or omit it, then whenever an item *x* is taken from the host filesystem and *x* is a symbolic link, `mkrcfsimg` resolves the symbolic link and includes its target file or directory. If you specify `-followlink`, `mkrcfsimg` includes the symbolic link itself in the image filesystem. It's up to you to include in the image whatever the link points to.

`gid` attribute

```
gid=id_spec
```

Set the group ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the group ID is taken from the host file; for an inline file, the group ID is the group of the user running `mkrcfsimg`. The default value for this attribute is `*`.

`label` attribute

```
label=string
```

Set a plain-text string (up to 32 bytes) to record in the superblock.

`mtime` attribute

```
mtime=time_spec
```

Set the timestamps of the files or directories to the specified time. The *time_spec* must be either:

- a single asterisk (`*`), meaning that the host file's timestamp should be used (the default behavior)

  or:

- in a format based on ISO8601:

  ```
  YYYY-MM-DD-HH:MM:SS
  ```

  You must provide all six elements. The time is always interpreted as UTC.

Timestamps specified with the `mtime` attribute aren't affected by the -n option.

`optional` attribute (boolean)

```
+|-optional
```

If true, and the host file can't be found, output a warning and continue building the embedded filesystem. If false, and the host file can't be found, output an error message and exit `mkrcfsimg`. The default is true.

Copyright © 2014, QNX Software Systems Limited

perms attribute

perms=*perm_spec*

Set the access permissions of the file. The *perm_spec* can be one of the following:

- a number (just as with the chmod command)
- an asterisk (*) to use the host file's permissions (or 0666 for inline files)
- a symbolic mode string to delete, add, or set permissions. This string is a subset of chmod's and consists of:

  1. a combination of u, g, o, and a
  2. one of -, =, or +
  3. a combination of r, w, x, s, g, and t.

  You can include more than one symbolic mode string, separating them with a comma (,).

The default is *.

When running on a Windows host, mkrcfsimg can't get the execute (x), setuid ("set user ID"), or setgid ("set group ID") permissions from the file. Use the perms attribute to specify these permissions explicitly. You might also have to use the *uid* (p. 1304) and *gid* (p. 1302) attributes to set the ownership correctly. To determine whether or not a utility needs to have the setuid or setgid permission set, see its entry in the *Utilities Reference*.

ELF executables and shared objects are automatically marked as executable (unless you specify [+raw]).

---

prefix attribute

prefix=*path*

Set the prefix on the target file names. The default is the empty string.

search attribute

search=*path*[:*path*...]

This attribute specifies that mkrcfsimg should search for the file in the named locations on the host system. The search directory portion of the host file name isn't included in the name that's stored in the RCFS. Colon separators and forward slashes in the paths are the standard Unix conventions, but for Windows searches, you must use the standard Windows conventions, such as semicolon separators and backslashes in the paths.

type attribute

type=*file_type*

Copyright © 2014, QNX Software Systems Limited

Set the type of the files being created in the image filesystem. Allowable types are:

- `link` — a symbolic link
- `fifo` — a named pipe
- `file` — a regular, everyday file (the default)
- `dir` — a directory

Specifying `[type=dir]` tells `mkrcfsimg` to make the named file a directory; you don't need to specify the type when you're copying the contents of a directory. For example, this command:

```
[type=dir]/usr/bin=/usr/nto/x86/bin
```

creates an *empty* directory named `/usr/bin`, with the same owner and permissions as for the host directory. To recursively copy `/usr/nto/x86/bin` to `/usr/bin`, you just need to specify:

```
/usr/bin=/usr/nto/x86/bin
```

`uid` attribute

`uid=`*`id_spec`*

Set the user ID number for the file. The value of this attribute may be either a number or an asterisk (`*`). If it's an asterisk, the user ID is taken from the host file; for an inline file, the user ID is the user running `mkrcfsimg`. The default value for this attribute is `*`.

`verifier` attribute

`verifier=`*`verifier_type`*

Enable hashing and set the type of verifier:

`inline`

RCFSv2 inline

`log`

Generate a file called `verifier.log` in the output filesystem, containing a set of SHA-256 hashes, one per file.

This attribute is always applied globally.

`zone` attribute

`zone=`*`number`*

Assign a zone number to the file or directory. Zones let you group files; the file data is physically arranged on disk by zone number. During mounting, all of the files in a

Copyright © 2014, QNX Software Systems Limited

given zone can be validated by doing a sequential read of the raw blocks encompassing the zone. If you use zones, your RCFS image must be cryptographically signed.

Exit status:

0

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

mkrec

Convert a binary image into ROM format (QNX)

Syntax:

```
mkrec [option | imagefilename] …
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-a *align*[K|M|G]

Force the next file to be aligned on an *align*-byte boundary (the default is 1 byte).

-f *format*

Output format:

- `binary` or `b` — raw binary file.
- `srec` or `s` — Motorola S records.
- `intel` or `i` — Intel hex records.
- `full` or `f` — pad the file to the size specified by the -s option (implies binary format).

The default is `srec`. When using `binary`, the file's offset is printed to *stderr*.

-l *reclen*

("el") Length of data bytes per line (the default is 32).

-o *offset*

The offset, in hexadecimal (the default is `0`).

-r

Suppress the reset vector record.

-s *romsize*[K|M|G]

Size of ROM (the default is `4G`). The case of the suffixes doesn't matter.

Copyright © 2014, QNX Software Systems Limited

Description:

The `mkrec` utility converts a binary image into either Motorola S records or Intel hex records, which are suitable input for most Flash/EPROM programmers. Most ICEs also expect these formats. The format defaults to Motorola S records but may be changed using the -f format option. Note that the Intel format is limited to 1M offsets.

By default, `mkrec` assumes the image contains code that is the initial program loader (IPL), which is connected to the power-on reset vector of the processor. The `mkrec` utility locates the image such that it's placed in the address space where it will end at the reset vector. A record is then output for the reset vector, which will do a 16-bit relative `jmp` to the start of the image.

The reset vector on an Intel (e.g. 486) processor is located at linear address `0xFFFFFFF0` (16 bytes below 4 Gigabytes). The offsets used for the records by default address this area. It's as though you had a 4-Gigabyte ROM and needed to load code into the top of it. This is what most ICEs expect. If you're sending your output to a small Flash/EPROM programmer, it may wish the offsets to be relocated to the top of the the Flash/EPROM alone. You can do this by using the -s *size* option.

If the -r option is set, it indicates that the image shouldn't be considered a power-on IPL. The record offsets for the image start at zero; a `jmp` isn't be programmed at the top of the device. You can use the -o option to change the record offset. Note that you can use the -o option only if you use -r.

For compatibility with most devices that accept these records, each record is limited to a maximum of 32 bytes. You can use the -l option to increase this limit to a maximum of 255 bytes. Larger records have less overhead and result in slightly faster downloads.

Examples:

The following converts the binary image `explr2` into Motorola S records. The offset used in the records starts the image such that it ends at the reset vector `0xFFFFFFF0`. A 16-bit relative `jmp` is programmed at `0xFFFFFFF0` to `jmp` to the start of the image.

```
mkrec explr2
```

The following converts the binary image `ipl` into Intel hex records. The offset used in the records starts the image such that it ends at the reset vector in the 256K ROM (`0x3FFF0`). A 16-bit relative `jmp` is programmed at `0x3FFF0` to `jmp` to the start of the image.

```
mkrec -f i -s 256k ipl
```

The following converts the binary image `notipl` into Motorola S records. The offset used in the records starts at `0`; a reset vector `jmp` isn't output.

```
mkrec -r notipl
```

Copyright © 2014, QNX Software Systems Limited

/etc/moduli

System moduli file for `sshd`

Name:

`/etc/moduli`

Description:

The `/etc/moduli` file contains the system-wide Diffie-Hellman prime moduli for `sshd`. For more information, see *`/etc/moduli`* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

more

Display files on a page-by-page basis (POSIX)

Syntax:

```
more [-ceisu] [-n number] [-p pattern]
     [-/ pattern] [-t tag] [-x tabstop] [file...]
```

Runs on:

QNX Neutrino

Options:

-/ pattern

Same as -p *pattern*.

-c

For each full screen of text that's displayed, clear the screen from the first line and display the next full screen of text.

-e

Stop after displaying the last line in the file. If the next command that displays text causes `more` to reach end-of-file again, `more` exits. If the file is shorter than a single screen, `more` exits at end-of-file regardless.

-i

Ignore case during searches. Uppercase and lowercase letters are considered identical.

-n *number*

Specify the number of lines that constitute a full screen of text. The *number* argument is a positive decimal integer. The -n option overrides any values obtained from the environment.

-p *pattern*

Search for a line that matches *pattern*. The current position is set to the first matched line. If no match is found, the first line in the file is the current position.

-s

Copyright © 2014, QNX Software Systems Limited

Replace consecutive empty lines with a single empty line.

-t *tag*

Display the file containing the tag named by the *tag* argument. Note that for this to work, the file `tags` must reside in the current directory.

-u

Always display backspaces as control characters (e.g. as the two-character sequence `^H`) and leave carriage-return/linefeed (`\r\n`) sequences alone.

By default, `more` makes special use of backspaces and carriage-return/linefeed (`\r\n`) sequences. If a backspace appears next to an underscore character, the character is displayed as underlined text, provided the terminal type supports underlined text. If a backspace appears between two identical characters, the first character is displayed as bold text, provided the terminal type supports bold text display.

-x *tabstop*

Set tabs at the positions specified by *tabstop*. The default is four spaces, unless the ***POSIX_STRICT*** environment variable is defined, in which case it's eight spaces.

file

A pathname of an input file. If no *file* operands are specified, `more` uses the standard input. If a *file* operand is the dash character (-), the standard input is read at that point of the sequence.

Description:

The `more` utility lets you view text files one screenfull at a time. The utility determines the number of lines that make a full screen by looking in the terminal database. However, you can use the ***LINES*** environment variable to override the value found in the database, and the -n option to override the ***LINES*** variable.

If the standard output isn't a terminal device, the number of lines that make up a full screen of text is considered to be infinite. In a pipeline, all input files are copied to the standard output in their entirety. On terminals, `more` displays text one screen at a time.

The `more` command can be very useful when another utility prints more information to the standard output than can be displayed on a single screen. By piping the output to `more`, you can scroll through the displayed output at leisure. For example:

```
ls | more
```


Copyright © 2014, QNX Software Systems Limited

pipes the output from the *ls* (p. 1139) command to `more`, allowing you to scroll through the output.

The `more` command is a link to *less* (p. 1100), which behaves according to the command name it was invoked as.

Environment variables:

EDITOR

The editor to use.

LINES

A decimal integer value to be used as the number of lines in a screenfull.

MORE

A string containing options described in the Options section of this utility, preceded by hyphens and separated by blank characters as on the command line. Command-line options override those specified in the ***MORE*** variable. The ***MORE*** variable takes precedence over the ***TERM*** and ***LINES*** variables.

TERM

The name of the terminal type.

POSIX_STRICT

Interpret options according to POSIX specifications.

Exit status:

0

Successful completion.

>0

An error occurred.

Contributing author:

Mark Nudelman

Copyright © 2014, QNX Software Systems Limited

mount

Mount a block special device or remote filesystem

Syntax:

```
mount [-abwruv] [-t type [-o options] [special] mountpoint]
mount [-abwruv] [-T type [-o options] special [mountpoint]]
mount [-abwruv] -e [-t|T type] [-o options] special
      [mountpoint]
mount
```

Runs on:

QNX Neutrino

Options:

-a

Mount all the devices listed in the */etc/fstab* (p. 846) file (or autodetected later on). If you also specify a *type*, mount only those entries. This option is ignored if you specify *special* or *mountpoint*.

-b

Prevent the lookup of the `fstab` file.

-e

Enumerate the children of the special device.

-o *options*

Options specific to the server doing the mounting. These options include:

- before — Mount the filesystem so that it's resolved before any other filesystems mounted at the same pathname (in other words, it's placed in front of any existing mount). When you access a file, the system looks on this filesystem first.
- after — Mount the filesystem so that it's resolved after any other filesystems mounted at the same pathname (in other words, it's placed behind any existing mount). When you access a file, the system looks on this filesystem last, and only if the file wasn't found on any other filesystems.

For more information, see “Ordering mountpoints” in the Process Manager chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

-r

Mount the device as read-only.

-T *type* ... *special* [*mountpoint*]

The special device is a string that may specify a real device or may be just a hint for the server. If *mountpoint* isn't specified, the server will automatically create an appropriate mountpoint.

-t *type* ... [*special*] *mountpoint*

If the optional *special* string is given, the mount request goes to the server which created, or is responsible for, the special device. If this special device does not exist, the server interprets the string as a hint. If *special* is not given, it is passed as `NULL`.

-u

Mount for update (remount).

-v

Increase the verbosity.

-w

Mount the device as read/write. This is the default (if the physical media permit).

mountpoint

Where the device is to be mounted on your system.

special

The name of the special device.

type

The type of filesystem or manager to mount:

type:	Filesystem or manager:
`cd`	*`fs-cd.so`* (p. 787)
`cifs`	*`fs-cifs`* (p. 790)
`dos`	*`fs-dos.so`* (p. 795)
`etfs`	Embedded Transaction Filesystem (e.g. *`fs-etfs-ram`* (p. 801))

Copyright © 2014, QNX Software Systems Limited

type:	Filesystem or manager:
ext2	*fs-ext2.so* (p. 807)
io-audio	*io-audio* (p. 989)
io-pkt	*io-pkt-v4*, *io-pkt-v4-hc*, *io-pkt-v6-hc* (p. 1007)
io-usb	*io-usb* (p. 1015)
mac	*fs-mac.so* (p. 809)
nfs	*fs-nfs2* (p. 811), *fs-nfs3* (p. 814)
nt	*fs-nt.so* (p. 818)
qnx4	*fs-qnx4.so* (p. 820)
qnx6	*fs-qnx6.so* (p. 823)
rcfs	*fs-rcfs.so* (p. 827)
udf	*fs-udf.so* (p. 829)

If you don't specify the filesystem, `mount` tries to determine which to use. If it can't figure out which to use, it uses `qnx4`.

Specify `io-pkt` for *type* no matter which of `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` you're mounting.

Description:

Without options, `mount` displays the current mountpoints. With options set, this utility mounts the block special device or remote filesystem, *special*, as the specified *mountpoint*. To mount a real special device, use the -t option; to specify a special-device string (which isn't necessarily a real device), use -T.

Some servers may not support all the `mount` options, especially with respect to remounting and enumerating.

The `mount` utility supports the */etc/fstab* (p. 846) file.

If you specify a filesystem option (e.g. `noatime`) on a block filesystem, and then you remount the filesystem (`mount -u`), the flag is ignored. The absence of the flag is interpreted as your asking for access time updates to be turned on. There's no way for the code in `io-blk` to determine if you wanted to use

Copyright © 2014, QNX Software Systems Limited

the default, and therefore didn't specify anything, or really did want access time updates to be turned on, and therefore didn't specify anything.

Similarly, if you mounted the filesystem as read-only and then remount it, the filesystem returns to its default setting.

To maintain the settings, specify the options again using the -o option for the mount command. For example:

```
mount -u -o noatime ...
```

Examples:

Mount a QNX 4 filesystem on a hard drive as `/mnt/fs`:

```
mount -t qnx4 /dev/hd0t77 /mnt/fs
```

Mount a device driver for `io-pkt*`. In this example, `devn-ne2000.so` is the name of the shared object that `io-pkt*` needs to load for the driver, not the name of a real device:

```
mount -T io-pkt devn-ne2000.so
```

If you want to pass options to the driver, use the -o option *before* the name of the shared object:

```
mount -T io-pkt -o mac=12345678 devn-ne2000.so
```

Enumerate the hard disk partition table:

```
mount -e /dev/hd0
```

This will re-read the disk partition table for `/dev/hd0`, and create, update or delete `/dev/hd0tXX` block-special files for each partition. This is used in the following two scenarios:

- when the disk driver is used without any automatic enumeration (blk auto=none), or
- when the partition table has been modified (for example, with *fdisk* (p. 734)).

Mount a CIFS filesystem (*fs-cifs* (p. 790) must be running first):

```
mount -T cifs -o abc,efg //node123:1.1.1.1:/C /ctest
```

Where your name is `abc`, your password is `efg`, your CIFS server is `node123` with an IP address of `1.1.1.1`, the share you want to mount is `/C`, and the mountpoint you want to use is `/ctest`.

Mount an NFS 2 client filesystem (*fs-nfs2* (p. 811) must be running first):

```
mount -T nfs 10.1.0.22:/home /mnt/home
```

Copyright © 2014, QNX Software Systems Limited

Mount an NFS 3 client filesystem (*fs-nfs3* (p. 814) must be running first):

```
mount -T nfs -o ver3 server_node:/qnx_bin /bin
```

Mount the Qnet network protocol:

```
mount -T io-pkt /lib/dll/lsm-qnet.so
```

Display the current mountpoints:

```
mount
```

Mount the shared object that supports Enhanced Host Controller Interface (EHCI) USB controllers:

```
mount -T io-usb devu-ehci.so /dev/io-usb/io-usb
```

Remount the filesystem that's currently mounted at / as read-only:

```
mount -ur /
```

Remount the filesystem that's currently mounted at / as read-write:

```
mount -uw /
```


Copyright © 2014, QNX Software Systems Limited

mq

Manage message queues (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
mq [options] &
```

Runs on:

QNX Neutrino

Options:

-d

Don't daemonize.

-m ***num_msgs***

Set the default for the maximum number of messages, for use if the *mq_attr* argument to *mq_open()* is `NULL`. The default is 64 messages.

-N ***path***

Set the pathname of the directory for message queues. The default is `/dev/mq`.

-s ***size***

Set the default message size, for use if the *mq_attr* argument to *mq_open()* is `NULL`. The default is 256 bytes.

-U ***user_name***

-U ***uid*****[:***gid*****[,***sup_gid*****]*]]**

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.

Copyright © 2014, QNX Software Systems Limited

- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

Description:

The `mq` manager implements POSIX 1003.1b message queues. When you create a queue, it appears in the pathname space under `/dev/mq`.

The `/dev/mq` directory doesn't appear until you actually create a queue.

You can change this directory to union over the directory exported by the `mqueue` server by using the `mq -N/dev/mqueue` option, but we don't recommend this, because it may cause some user-namespace confusion.

This implementation uses the kernel's *asynchronous messaging* facility to buffer the messages within the kernel itself, and eliminates the context-switching overheads of using an external server (i.e. `mqueue`) in each message-queue operation, thus greatly improving the performance of POSIX message queues.

In order to use the `mq` implementation, you must link your application(s) against the `libmq` library. In a manual build, specify the `-l mq` option; in automatic/recursive builds, use this setting in your `common.mk` file:

```
LIBS += mq
```

For more information, see the Managing POSIX Message Queues technote.

Copyright © 2014, QNX Software Systems Limited

mqueue

Manage message queues (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
mqueue [-d] [-p priority] &
```

Runs on:

QNX Neutrino

Options:

-d

Don't daemonize.

-p *priority*

Run at the given static priority.

Description:

The `mqueue` manager implements POSIX 1003.1b message queues. When you create a queue, it appears in the pathname space under `/dev/mqueue`.

The `/dev/mqueue` directory doesn't appear until you actually create a queue.

The traditional POSIX message queue manager is a resource manager that uses messages to communicate with its clients. You can access it locally or remotely, allowing for network-wide message queues. On a QNX Neutrino system, the default `sysinit` script starts `mqueue`.

Starting with release 6.3.0, *`procnto`* (p. 1586) manages named semaphores, which `mqueue` used to do (`mqueue` still manages named semaphores if it detects that `procnto` isn't doing so).

For more information, see the Managing POSIX Message Queues technote.

Copyright © 2014, QNX Software Systems Limited

mrouted

IP multicast routing daemon

Syntax:

```
mrouted [-c config_file] [-d [debug_level]] [-p]
```

Runs on:

QNX Neutrino

Options:

-c *config_file*]

Specify a configuration file (default is `/etc/mrouted.conf`).

-d [*debug_level*]

Specify debug level (default is 0).

-p

Start `mrouted` in a non-pruning mode.

Description:

The `mrouted` utility is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP) (*RFC 1075* specifies an earlier version of this protocol). The utility maintains topological knowledge via a distance-vector routing protocol (like RIP, described in *RFC 1058*), where it implements a multicast datagram forwarding algorithm called Reverse Path Multicasting.

The `mrouted` utility forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. The multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it doesn't extend beyond those subnetworks that have members of the destination group. Hence, datagrams aren't forwarded along those branches which have no listeners of the multicast group. The IP time-to-live of a multicast datagram can be used to limit the range of multicast datagrams.

In order to support multicasting among subnets that are separated by (unicast) routers that don't support IP multicasting, `mrouted` includes support for “tunnels”, which are virtual point-to-point links between pairs of `mrouted`s located anywhere in an internet. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. The

Copyright © 2014, QNX Software Systems Limited

encapsulation is added on entry to a tunnel, and stripped off on exit from a tunnel. By default, the packets are encapsulated using the IP-in-IP protocol (IP protocol number 4). Older versions of `mrouted` tunnel using IP source routing, which puts a heavy load on some types of routers.

This version doesn't support IP source route tunneling.

The tunneling mechanism allows `mrouted` to establish a virtual internet, for the purpose of multicasting only, which is independent of the physical internet, and which may span multiple Autonomous Systems. This capability is intended for experimental support of internet multicasting only, pending widespread support for multicast routing by the regular (unicast) routers. The `mrouted` utility suffers from the well-known scaling problems of any distance-vector routing protocol, and doesn't (yet) support hierarchical multicast routing.

The `mrouted` utility handles multicast routing only; there may or may not be unicast routing software running on the same machine as `mrouted`. With the use of tunnels, it isn't necessary for `mrouted` to have access to more than one physical subnet in order to perform multicast forwarding.

Invocation

If -d isn't specified, or if the debug level is specified as 0, `mrouted` detaches from the invoking terminal. Otherwise, it remains attached to the invoking terminal and responsive to signals from that terminal. If -d is specified with no argument, the debug level defaults to 2. Regardless of the debug level, `mrouted` always writes warning and error messages to the system log daemon. Nonzero debug levels have the following effects:

Level 1

All messages that are sent to the system log are also printed to *stderr*. In order to capture the log messages, you need to have *`syslogd`* (p. 1885) running.

Level 2

All level 1 messages plus notifications of significant events are printed to *stderr*.

Level 3

All level 2 messages plus notifications of all packet arrivals and departures are printed to *stderr*.

At startup, `mrouted` writes its *pid* to the file `/var/run/mrouted.pid`.

Configuring `mrouted`

Copyright © 2014, QNX Software Systems Limited

The `mrouted` utility automatically configures itself to forward on all multicast-capable interfaces, that is, interfaces that have the `IFF_MULTICAST` flag set (excluding the loopback "interface"), and it finds other `mrouted`s directly reachable via those interfaces. To override the default configuration, or to add tunnel links to other `mrouted`s, configuration commands may be placed in `/etc/mrouted.conf` (or an alternative file, specified by the -c option). There are five types of configuration commands:

`phyint` ***local-addr*** **[disable] [metric** ***m*****] or [threshold** ***t*****] [rate_limit** ***b*****] or [boundary (*****boundary-name*****|*****scoped-addr*****/*****mask-len*****)] or [altnet** ***network*****/*****mask-len*****]**

This command can be used to disable multicast routing on the physical interface identified by local IP address *local-addr*, or to associate a nondefault metric or threshold with the specified physical interface. The local IP address *local-addr* may be replaced by the interface name (e.g `le0`). If a `phyint` is attached to multiple IP subnets, describe each additional subnet with the altnet keyword. The `phyint` configuration commands must precede `tunnel` commands.

`tunnel` ***local-addr remote-addr*** **[metric** ***m*****] or [threshold** ***t*****] [rate_limit** ***b*****] or [boundary (*****boundary-name*****|*****scoped-addr*****/*****mask-len*****)]**

This command can be used to establish a tunnel link between local IP address *local-addr* and remote IP address *remote-addr*, and to associate a nondefault metric or threshold with that tunnel. The local IP address *local-addr* may be replaced by the interface name (e.g. `le0`). The remote IP address *remote-addr* may be replaced by a hostname, if and only if the hostname has a single IP address associated with it. The `tunnel` must be set up in the `mrouted.conf` files of both routers before it can be used.

`cache_lifetime` ***ct***

The time (in seconds) that a cached multicast route stays in kernel before timing out (the default is 300). The value of this entry should lie between 300 (5 min) and 86400 (1 day).

`pruning` ***off/on***

Allow `mrouted` to act as a non-pruning router (default mode is pruning-enabled). It's also possible to start `mrouted` in a non-pruning mode using the -p option on the command line. It's expected that a router would be configured in this manner for test purposes only.

`name` ***boundary-name scoped-addr/mask-len***

Copyright © 2014, QNX Software Systems Limited

You may assign names to boundaries with this keyword, to make configuration easier. The boundary option on `phyint` or `tunnel` commands can accept either a name or a boundary.

The file format is free-form; whitespace (including newlines) isn't significant.

Configuration command options

The boundary and altnet options may be specified as many times as necessary.

metric

The "cost" associated with sending a datagram on the given interface or tunnel; it may be used to influence the choice of routes (default is 1). Metrics should be kept as small as possible, because `mrouted` cannot route along paths with a sum of metrics greater than 31.

threshold

The minimum IP time-to-live required for a multicast datagram to be forwarded to the given interface or `tunnel` (default is 1). It's used to control the scope of multicast datagrams. (The TTL of forwarded packets is only compared to the threshold, it isn't decremented by the threshold.) Every multicast router decrements the TTL by 1.

In general, all `mrouted`s connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or `tunnel`.

rate_limit

Allows the network administrator to specify a certain bandwidth in Kbits/second that would be allocated to multicast traffic. It defaults to 500Kbps on `tunnel`s, and 0 (unlimited) on physical interfaces.

boundary

Allows an interface to be configured as an administrative boundary for the specified scoped address. Packets belonging to this address won't be forwarded on a scoped interface. The boundary option accepts either a name or a boundary spec.

The `mrouted` utility won't initiate execution if it has fewer than two enabled virtual interfaces (*vif*s), where a *vif* is either a physical multicast-capable interface or a `tunnel`. It'll log a warning if all of its *vif*s are tunnels; such an `mrouted` configuration would be better replaced by more direct tunnels (i.e. eliminate the middle man).

Signals

The `mrouted` utility responds to the following signals:

SIGHUP

Copyright © 2014, QNX Software Systems Limited

Restarts `mrouted`. The configuration file is reread every time this signal is evoked.

SIGINT

Terminates execution gracefully (i.e. by sending goodbye messages to all neighboring routers).

SIGTERM

Same as `SIGINT`.

SIGUSR1

Dumps the internal routing tables to `/var/tmp/mrouted.dump`.

SIGUSR2

Dumps the internal cache tables to `/var/tmp/mrouted.cache`.

SIGQUIT

Dumps the internal routing tables to *stderr* (only if `mrouted` was invoked with a nonzero debug level).

For convenience in sending signals, on startup, `mrouted` writes its *pid* to `/var/run/mrouted.pid`.

Based on:

RFC 1075, which specifies an earlier version of DVMRP.

DVMRP is described, along with other multicast routing algorithms, in the paper *Multicast Routing in Internet-works and Extended LANs* by S. Deering, in the Proceedings of the ACM SIGCOMM '88 Conference.

Examples:

This is an example configuration for a mythical multicast router at a big school.

```
# mrouted.conf example
#

# Name our boundaries to make it easier.

name LOCAL 239.255.0.0/16
name EE 239.254.0.0/16

# le1 is our gateway to compsci, don't forward our
# local groups to them.

phyint le1 boundary EE

# le2 is our interface on the classroom net, it has four
# different length subnets on it. Note that you can use
# either an ip address or an interface name.

phyint 172.16.12.38 boundary EE altnet 172.16.15.0/26
altnet 172.16.15.128/26 altnet 172.16.48.0/24
```


Copyright © 2014, QNX Software Systems Limited

```
# atm0 is our ATM interface, which doesn't properly
# support multicasting.

phyint atm0 disable

# This is an internal tunnel to another EE subnet
# Remove the default tunnel rate limit, since this
# tunnel is over Ethernets

tunnel 192.168.5.4 192.168.55.101 metric 1 threshold 1
rate_limit 0

# This is our tunnel to the outside world.
# Careful with those boundaries, Eugene.

tunnel 192.168.5.4 10.11.12.13 metric 1 threshold 32
boundary LOCAL boundary EE
```

Routing tables

The routing tables look like this:

```
Virtual Interface Table
 Vif  Local-Address                  Metric  Thresh  Flags
  0   36.2.0.8      subnet: 36.2         1       1    querier
                    groups: 224.0.2.1
                            224.0.0.4
                  pkts in: 3456
                 pkts out: 2322323

  1   36.11.0.1     subnet: 36.11        1       1    querier
                    groups: 224.0.2.1
                            224.0.1.0
                            224.0.0.4
                  pkts in: 345
                 pkts out: 3456

  2   36.2.0.8      tunnel: 36.8.0.77    3       1
                     peers: 36.8.0.77 (2.2)
                boundaries: 239.0.1
                          : 239.1.2
                  pkts in: 34545433
                 pkts out: 234342

  3   36.2.0.8      tunnel: 36.6.8.23    3       16

Multicast Routing Table (1136 entries)
 Origin-Subnet    From-Gateway    Metric Tmr In-Vif  Out-Vifs
 36.2                                1    45    0    1* 2  3*
 36.8             36.8.0.77          4    15    2    0* 1* 3*
 36.11                               1    20    1    0* 2  3*
 .
 .
 .
```

In the above example, there are four *Vif*s connecting to two subnets and two tunnels. The *vif* 3 tunnel isn't in use (no peer address). The *vif* 0 and *vif* 1 subnets have some groups present; tunnels never have any groups. This instance of `mrouted` is the one responsible for sending periodic group membership queries on the *vif* 0 and *vif* 1 subnets, as indicated by the "querier" flags. The list of boundaries indicate the scoped addresses on that interface. A count of the number of incoming and outgoing packets is also shown at each interface.

Associated with each subnet from which a multicast datagram can originate is the address of the previous hop router (unless the subnet is directly-connected), the metric of the path back to the origin, the amount of time since we last received an update for this subnet, the incoming *vif* for multicasts from that origin, and a list of outgoing *vif*s. A * means that the outgoing *vif* is connected to a leaf of the broadcast tree rooted

Copyright © 2014, QNX Software Systems Limited

at the origin, and a multicast datagram from that origin is forwarded on that outgoing *vif* only if there are members of the destination group on that leaf.

The `mrouted` utility also maintains a copy of the kernel forwarding cache table. Entries are created and deleted by `mrouted`.

The cache tables look like this:

```
Multicast Routing Cache Table (147 entries)
 Origin             Mcast-group    CTmr  Age Ptmr IVif Forwvifs
 13.2.116/22        224.2.127.255    3m   2m    -  0    1
>13.2.116.19
>13.2.116.196
 138.96.48/21       224.2.127.255    5m   2m    -  0    1
>138.96.48.108
 128.9.160/20       224.2.127.255    3m   2m    -  0    1
>128.9.160.45
 198.106.194/24     224.2.135.190    9m   28s  9m  0P
>198.106.194.22
```

Each entry is characterized by the origin subnet number and mask and the destination multicast group.

>

A “>” as the first character on an additional line is printed for each source on the subnet. There can be many sources in one subnet.

Age

The time since this cache entry was originally created. Since cache entries get refreshed if traffic is flowing, routing entries can grow very old.

CTmr

Indicates the lifetime of the entry. The entry is deleted from the cache table when the timer decrements to zero.

Forwvifs

Shows the interfaces along which datagrams belonging to the source-group are forwarded. A `p` indicates that no datagrams are being forwarded along that interface. An unlisted interface is a leaf subnet with no members of the particular group on that subnet. A `b` on an interface indicates that it's a boundary interface, i.e. traffic won't be forwarded on the scoped address on that interface.

Ivif

Indicates the incoming *vif* for multicast packets from that origin. Each router also keeps a record of the number of prunes received from neighboring routers for a particular source and group. If there are no members of a multicast group on any downward link of the multicast tree for a subnet, a

Copyright © 2014, QNX Software Systems Limited

prune message is sent to the upstream router. They're indicated by a `P` after the *vif* number.

Ptmr

The amount of time until the upstream prune times out, or simply a dash if no prune was sent.

Files:

`/etc/mrouted.conf`

Default configuration file for `mrouted`.

`/var/run/mrouted.pid`

At startup, `mrouted` writes its *pid* to this file.

`/var/tmp/mrouted.dump`

The `SIGUSR1` signal causes `mrouted` to dump the internal routing tables to this file.

`/var/tmp/mrouted.cache`

The `SIGUSR2` signal causes `mrouted` to dump the internal cache tables to this file.

Contributing author:

Steve Deering, Ajit Thyagarajan, Bill Fenner.

License:

The `mrouted` program is COPYRIGHT 1989 by The Board of Trustees of Leland Stanford Junior University; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

mv

Move files (POSIX)

Syntax:

```
mv [-f|-i] [-v|-V] source_file target_file
mv [-f|-i] [-v|-V] source_file... target_dir
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-f

Force an overwrite; don't prompt for confirmation if the destination path exists. Overwrite even read-only files.

-i

Run interactively; write a prompt to the standard error output before moving any file that would overwrite an existing file. If confirmation is received, overwrite the existing file. Otherwise, go on to the next file.

-V

(QNX Neutrino extension) Be very verbose.

-v

(QNX Neutrino extension) Be verbose.

source_file

The pathname of a file or directory to be moved.

target_file

The new pathname of the file or directory to be moved.

target_dir

The pathname of an existing directory that the source file is to move to.

Description:

The `mv` command has two syntax forms:

Copyright © 2014, QNX Software Systems Limited

mv [-fl-i] [-vl-V] ***source_file target_file***

The mv utility moves *source_file* to the destination specified by *target_file*. This first syntax form is assumed when the final operand you specify doesn't name an existing directory.

mv [-fl-i] [-vl-V] ***source_file... target_dir***

The mv utility moves each *source_file* file to a destination file in the directory named by *target_dir*. The destination's filename under the target directory is the same as its basename (final path component).

For example:

```
mv dir/dir/myfile /existingdir
```

moves `dir/dir/myfile` to `existingdir/myfile`.

This second syntax form is assumed when either the destination names an existing directory, or when more than one source file is specified.

By default, mv overwrites an existing file without warning or confirmation whenever you have write permission on the file.

The mv utility asks you for confirmation if the following conditions are met:

- you haven't specified the -f option
- you lack write permission
- the standard input is a terminal

Upon receiving confirmation, mv overwrites the target file. It can do this only if you own the file or you're a superuser.

If you want mv to request confirmation before overwriting any file, specify the -i (interactive) option. If you want mv to overwrite whenever possible without asking for confirmation, specify the -f (force) option.

As long as the input files specified by each *source_file* are on the same device as the target, the *source_file* operand can be of any file type. If the source and target reside on different devices, the *source_file* is copied to the target and then removed. If the *source_file* is a directory, this means that any FIFO or character special files under the original directory aren't copied. Since the copy isn't 100% successful, the original *source_file* isn't removed.

Examples:

In the current directory, rename the file `orange` to banana.

```
mv orange banana
```

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

All input files were moved successfully.

>0

An error occurred.

Caveats:

If the copying of a directory is prematurely terminated by a signal or error, `mv` may leave a partial copy of the directory at the destination. In this case, the directory tree at *source_file* isn't modified.

When the *source_file* and *target_file* are on different filesystems (i.e. not on the same mounted partition), the `mv` utility spawns the *cp* (p. 141) utility to copy the file(s), and if the `cp` succeeds, spawns the *rm* (p. 1673) utility to remove the originals.

Copyright © 2014, QNX Software Systems Limited

Chapter 15
N

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`ntoarmv7-*`, `ntox86-*`	*`addr2line`* (p. 34), *`ar`* (p. 45), *`c++filt`* (p. 84), *`gcc`* (p. 889), *`gcov`* (p. 890), *`gdb`* (p. 892), *`gprof`* (p. 913), *`g++`* (p. 889), *`ld`* (p. 1096), *`nm`* (p. 1358), *`objcopy`* (p. 1406), *`objdump`* (p. 1407), *`ranlib`* (p. 1657), *`readelf`* (p. 1660), *`size`* (p. 1771), *`strings`* (p. 1861), *`strip`* (p. 1862)

The following have been deprecated:

Instead of:	Use:
`named-xfer`	Obsolete; `named` no longer uses it
`nfm-autoip.so`	*`lsm-autoip.so`* (p. 1145)
`npm-pppmgr.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-pppoe.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-qnet-compat.so`	No replacement
`npm-qnet-l4_lite.so`	*`lsm-qnet.so`* (p. 1149)
`npm-tcpip-v4.so`, `npm-tcpip-v6.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-ttcpip.so`	No replacement

This chapter describes the utilities, etc. whose names start with “N”.

Copyright © 2014, QNX Software Systems Limited

named

Internet domain name server

Syntax:

```
named [-4] [-6] [-c config-file] [-d debug-level] [-f] [-g]
      [-m flag] [-n num_cpus] [-p port] [-s] [-t directory]
      [-u user] [-v] [-x cache-file]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?named++NetBSD-5.0* in the NetBSD documentation.

Description:

The `named` daemon is a Domain Name System (DNS) server, part of the BIND 9 distribution from ISC. For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, `named` reads the default configuration file `/etc/named.conf`, read any initial data, and listen for queries.

Based on:

RFC 882, *RFC 883*, *RFC 973*, *RFC 974*, *RFC 1033*

Files:

`/etc/named.conf`

Default name server configuration file.

`/etc/namedb/named.pid`

The process ID.

Copyright © 2014, QNX Software Systems Limited

named-checkconf

Tool for checking the syntax of a named configuration file

Syntax:

```
named-checkconf [-h] [-v] [-j] [-t directory] {filename}
                [-z]
```

Runs on:

QNX Neutrino

Options:

See
http://netbsd.gw.com/cgi-bin/man-cgi?named-checkconf++NetBSD-5.0
in the NetBSD documentation.

Description:

The named-checkconf utility checks the syntax, but not the semantics, of a named configuration file. For more information, see
http://netbsd.gw.com/cgi-bin/man-cgi?named-checkconf++NetBSD-5.0
in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

named-checkzone, named-compilezone

Tools for converting and checking a zone file

Syntax:

```
named-checkzone [-d] [-h] [-j] [-q] [-v] [-c class]
                [-f format] [-F format] [-i mode]
                [-k mode] [-m mode] [-M mode] [-n mode]
                [-o filename] [-s style] [-S mode]
                [-t directory] [-w directory] [-D] [-W mode]
                {zonename} {filename}

named-compilezone [-d] [-j] [-q] [-v] [-c class]
                  [-C mode] [-f format] [-F format]
                  [-i mode] [-k mode] [-m mode] [-n mode]
                  [-o filename] [-s style] [-t directory]
                  [-w directory] [-D] [-W mode] {zonename}
                  {filename}
```

Runs on:

QNX Neutrino

Options:

See
http://netbsd.gw.com/cgi-bin/man-cgi?named-checkzone++NetBSD-5.0
in the NetBSD documentation.

Description:

The `named-checkzone` utility checks the syntax and integrity of a zone file. It performs the same checks as `named` does when loading a zone.

The `named-compilezone` utility is similar to `named-checkzone`, but it always dumps the zone contents to a specified file in a specified format. Additionally, it applies stricter check levels by default, since the dump output will be used as an actual zone file loaded by `named`.

For more information, see
http://netbsd.gw.com/cgi-bin/man-cgi?named-checkzone++NetBSD-5.0
in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

/etc/named.conf

Configuration file for named

Name:

/etc/named.conf

Description:

The /etc/named.conf file is the configuration file for *named* (p. 1332). For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?named.conf++NetBSD-5.0* in the NetBSD documentation.

See also:

- *TCP/IP Network Administration*
- *DNS and BIND* by Paul Albitz and Cricket Liu, O'Reilly & Associates (ISBN 1-56592-010-4)

Copyright © 2014, QNX Software Systems Limited

ndp

Control the IPv6 Neighbor Discovery Protocol

Syntax:

```
ndp -A wait [-ntl]
ndp -a [-ntl]
ndp -c [-nt]
ndp -d [-nt] hostname
ndp -f [-nt] filename
ndp -H
ndp -I [delete|interface]
ndp -i interface [flags...]
ndp -P
ndp -p
ndp -R
ndp -r
ndp -s [-nt] nodename ether_addr [temp] [proxy]
```

Runs on:

QNX Neutrino

Options:

-A *wait*

Repeat -a (dump NDP entries) every *wait* seconds.

-a

Dump the existing NDP entries.

-c

Erase all NDP entries.

-d *hostname*

Delete the NDP entry for this *hostname*.

-f *filename*

Parse the file specified by *filename*.

-H

Harmonize consistency between the routing table and the default router list; install the top entry of the list into the kernel routing table.

-I [delete*interface*]

Copyright © 2014, QNX Software Systems Limited

Show or specify the default interface that's to be used as the default route when there's no default router.

Option:	Action:
-I	Show the current default interface.
-I delete	Delete the current interface from the kernel.
-I *interface*	Use this interface as the default.

-i *interface* [*flags...*]

View the neighbor discovery (ND) information for this interface. If *flags* is specified, set or clear the flag(s) associated with this interface. A flag is cleared if it starts with the special character "-". Possible flags are:

nud

Toggle NUD (Neighbor Unreachability Detection) on the interface (default is on).

-l

Don't truncate the numeric IPv6 addresses.

-n

Don't try to resolve a numeric address to a hostname.

-P

Flush all the entries in the prefix list.

-p

Show the prefix list.

-R

Flush all the entries in the default router list.

-r

Show the default router list.

-s *nodename ether_addr* [temp] [proxy]

Register a permanent NDP entry for this node (use temp to register a temporary entry). If you specify proxy, the system acts as a proxy NDP server

Copyright © 2014, QNX Software Systems Limited

that responds to requests for *hostname* even though the host address is not its own.

-t

Print a timestamp on each entry so that it's possible to merge the output with `tcpdump`. Typically used with -A.

Description:

This utility manipulates the address mapping table used by the Neighbor Discovery Protocol (NDP).

Exit status:

0

Successful completion.

Nonzero

An error occurred.

Copyright © 2014, QNX Software Systems Limited

netstat

Show network status

Syntax:

Display a list of active sockets for each protocol:

```
netstat [-AanT] [-f address_family]
```

Display the contents of one of the other network data structures:

```
netstat [-dgiLmnrsTv] [-f address_family]
```

Continuously display (as per the *wait* interval) the information regarding packet traffic on the configured network interfaces:

```
netstat [-dnT] [-I interface] [-w wait]
```

Display statistics about the named protocol:

```
netstat [-T] [-p protocol]
```

Display statistics about the named protocol located at the virtual address:

```
netstat [-T] [-p protocol] -P pcbaddr
```

Display per-interface statistics for the specified protocol:

```
netstat [-p protocol] [-iT] [-I interface]
```

Display per-interface statistics for the specified address family:

```
netstat [-sT] [-f address_family] [-i] [-I interface]
```

Runs on:

QNX Neutrino

Options:

-A

Show the addresses of any protocol control blocks associated with sockets.

-a

Show the state of all sockets. Without -a, sockets used by server processes aren't shown.

-d

Copyright © 2014, QNX Software Systems Limited

Show the number of dropped packets.

-f *address_family*

Limit the statistics or address control block reports to those of the specified address family.

Address family:	*address_family* value:
AF_INET	inet
AF_INET6	inet6
AF_LOCAL	local or unix
AF_ARP	arp

-g

Show information related to multicast (group address) routing. By default, show the IP Multicast virtual-interface and routing tables. If -s is also specified, show the multicast routing statistics.

-I [*interface*]

If used with -w, show information about the specifed interface only. See "*Specifying an interface* (p. 1344)" for more information.

If used with -f *address_family* and -s, or with -p *protocol*, show per-interface statistics on the interface for *address_family* or *protocol*, respectively.

-i

Show the state of interfaces that have been auto-configured. Interfaces statically configured into a system but not located at boot time aren't shown.

If you also specify -a, show multicast addresses currently in use for each Ethernet interface and for each IP interface address. Multicast addresses are shown on separate lines following the interface address with which they're associated.

If used with -f *address_family* and -s, or with -p *protocol*, show per-interface statistics on the interface for *address_family* or *protocol*, respectively.

-L

Don't show link-level routes (e.g., IPv4 ARP or IPv6 neighbour cache).

-m

Copyright © 2014, QNX Software Systems Limited

Show statistics recorded by the memory-management routines (the network manages a private pool of memory buffers).

-n

Show network addresses as numbers (normally `netstat` interprets addresses and attempts to display them symbolically).

-P ***pcbaddr***

Dump the contents of the protocol control block (PCB) located at kernel virtual address *pcbaddr*. This address may be obtained using -A. The default protocol is TCP, but may be overridden with -p.

-p ***protocol***

Show statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the file *`/etc/protocols`* (p. 1595). A null response typically means that there are no interesting numbers to report. The utility complains if *protocol* is unknown or if there's no statistics routine for it.

-r

Show the routing tables. If -s is also specified, show the routing statistics instead.

-s

Show per-protocol statistics. If this option is repeated, counters with a value of zero are suppressed.

-T

Use TCP for name lookups (the default is UDP).

-v

Show extra (verbose) detail for the routing tables (-r), or avoid truncating long addresses.

-w ***wait***

Specify the time interval for displaying network interface statistics.

Description:

The `netstat` utility displays the contents of various network-related data structures.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *`mkefs`* (p. 1209), *`mketfs`* (p. 1219), or *`mkifs`* (p. 1241) on a Windows

host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Default display

The default display for active sockets shows the local and remote addresses, the send and receive queue sizes (in bytes), the protocol, and the internal state of the protocol. If a socket's address specifies a network but no specific host address, address formats are of the form *host.port* or *network.port*

When known, the host and network addresses are displayed symbolically according to */etc/hosts* (p. 946) and */etc/networks* (p. 1345), respectively. If a symbolic name for an address is unknown, or if -n is specified, the address is printed numerically, according to the address family. For more information regarding the Internet "dot format," see the *inet_*()* functions. Unspecified or wildcard addresses and ports appear as *.

Interface display

The interface display (-i or -w) provides a table of cumulative statistics regarding errors, collisions, and packets transferred. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed.

Routing table display

The routing table display (-r) indicates the available routes and the status of each. Each route consists of a destination host or network and a gateway to use in forwarding packets. The display includes several fields:

flags

Shows the state of the route.

Valid flags:	Constant:	Description;
1	RTF_PROTO2	Protocol specific routing flag #1
2	RTF_PROTO1	Protocol specific routing flag #2
B	RTF_BLACKHOLE	Discard packets during updates
C	RTF_CLONING	Generate new routes on use This is used by ARP to create the host-specific routes of

Copyright © 2014, QNX Software Systems Limited

Valid flags:	Constant:	Description;
		the hosts on the Ethernet. See option L.
D	RTF_DYNAMIC	The route was created dynamically by a redirect
G	RTF_GATEWAY	The route uses a gateway
H	RTF_HOST	The destination is a host (otherwise, it's a net)
L	RTF_LLINFO	The route contains a link-layer address. The host routes that ARP clones from the Ethernet network routes all have the link flag set.
M	RTF_MODIFIED	The route has been modified by a redirect
R	RTF_REJECT	Host or net unreachable
S	RTF_STATIC	The route has been added manually
U	RTF_UP	The route is up
X	RTF_XRESOLVE	External daemon translates the protocol to a link address

gateway

Direct routes are created for each interface attached to the local host. For such entries, this field shows the address of the outgoing interface.

interface

Indicates the network interface used for the route.

mtu

Shows the maximum transmission unit (MTU) associated with that route. This value is used as the basis for the TCP maximum segment size. The L

Copyright © 2014, QNX Software Systems Limited

flag appended to the MTU value indicates that the value is locked, and that path MTU discovery is turned off for that route. A - indicates that the MTU for this route hasn't been set, and a default TCP maximum segment size will be used.

refcnt

Gives the current number of active uses of the route. Connection-oriented protocols normally hold on to a single route for the duration of a connection; connectionless protocols obtain a route while sending to the same destination.

use

Provides a count of the number of packets sent using the route.

Specifying an interface

When -w is specified, `netstat` displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during auto-configuration) and a column summarizing information for all interfaces. Using -I, you can replace the primary interface with another interface. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

Caveats:

The -A, -a, -d, and -s options require the presence of the `/etc/protocols` file.

Copyright © 2014, QNX Software Systems Limited

/etc/networks

Network name database

Name:

/etc/networks

Description:

The networks file contains information regarding the known networks that make up the DARPA Internet. For each network, a single line should be present with the following information:

```
official_network_name  network_number  aliases
```

Items are separated by any number of spaces or tabs, or both. A pound sign (#) in a line indicates the beginning of a comment — any characters after a #, up to the end of the line, aren't interpreted by routines that search the file.

The *network number* may be specified in the conventional "." (dot) notation using the *inet_network()* routine from the internet address manipulation functions, *inet_*()*. Network names may contain any printable character other than a field delimiter, newline, or comment character.

Copyright © 2014, QNX Software Systems Limited

newgrp

Change to a new group

Syntax:

```
newgrp [-l|-s] [group]
```

Runs on:

QNX Neutrino

Options:

-l

("el") Change the environment to what would be expected if the user actually logged in again.

-s

Set the group ID of the parent; don't *exec()* a login shell.

group

A group name.

Description:

The `newgrp` utility starts a new shell with a new real and effective group ID. The user remains logged in and the current directory doesn't change, but file access permissions are based on the new real and effective group IDs.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

If no operands are specified, `newgrp` changes the group ID back to the group identified in the invoking user's entry in `/etc/passwd`.

If you don't use the -l option, the environment variables remain unchanged.

Note that the QNX Neutrino RTOS doesn't support group passwords. Only the user `root` may change the group ID to a group that it doesn't belong to.

Copyright © 2014, QNX Software Systems Limited

Files:

/etc/group

Contains information on the group IDs in the system. See passwd's "Files" section for a brief description of the format of this file.

/etc/passwd

Contains information on the user IDs in the system.

See the passwd utility "Files" section for a brief description of the format of these files.

Copyright © 2014, QNX Software Systems Limited

nfsd

NFS v2 & v3 and MOUNT v1 & v3 protocol server

You must be `root` to start this daemon.

Syntax:

```
nfsd [-DFPt] [-c file] [-f n] [-H n] [-h n] [-o option] [-p n]
     [-s n] [-x n] &
```

Runs on:

QNX Neutrino

Options:

-c *file*

Use *file* as the exports file. The default is */etc/exports* (p. 721).

-D

Operate in debugging mode.

-F

Truncate the `subdirs` and `mntdtab` files, and then exit.

-f *n*

Set the size of the open file cache (the default is 16).

The open file cache is used to cache open files and directories (with a 5-second idle timeout). If you know `nfsd` services only one client that only reads/writes to a single file, reducing this cache may save memory. If you know `nfsd` services many clients that read/write many files, increasing this cache could improve performance for read/write operations.

Keep this cache a reasonable size, as file descriptors (open files) are a limited resource — by default, QNX Neutrino sets a maximum of 1000 open files per process. Besides this cache, `nfsd` needs file descriptors for sockets (servicing TCP consumes more than just UDP) and internal *readdir()* operations.

Copyright © 2014, QNX Software Systems Limited

-H *n*

Specify the size of the file handle cache hash (the default is 997).

-h *n*

Specify the size of the file handle cache (the default is 200).

The file handle cache is a straight memory/performance trade-off, however it doesn't significantly affect read/write performance. It helps speed up `ls`-type operations (very useful for compiling/makefiles). To get a rough idea of how large this cache should (optimally) be, use the output of:

```
find mnt1 ... mntN | wc -l
```

-o *option*

Specify an additional option, where *option* is one of the following:

- nfsvers=2 — support NFS v2 only; the default is to support both NFS v2 and NFS v3.
- run=foreground — run in the foreground; don't fork.

-P

Parse the exports file, to check for errors, and then exit.

-p *n*

Run `nfsd` on port *n*, and don't register with `portmap`. By default, the port is 2049, and `nfsd` registers with `portmap`.

-s *n*

Flush the cache every *n* idle seconds (the default is 5).

-t

Service TCP transport.

-x *n*

Specify the size of the XID cache (default is 16).

The XID cache isn't used for performance, but rather to ensure nonidempotent operations are responded to correctly.

Consider what happens when a client issues a remove request. Normally, the server receives the request, removes the file, and sends back a successful response. Suppose that, for some reason, the server doesn't respond fast

Copyright © 2014, QNX Software Systems Limited

enough for the client, and the client retransmits the request. If the server tries to remove the file (again), it fails.

Instead, each request is assigned a transaction identifier, known as an *xid*, which remains constant for retransmissions. If the client retransmits the request, the server matches it with the previous request and just replies with the previous status. Generally, the busier the network and server are, the more requests are retransmitted by the client(s), and the larger the XID cache should be.

Description:

The `nfsd` daemon services both NFS mount requests and NFS requests, as specified by the exports file. When it starts, `nfsd` reads the `/etc/exports.`*hostname* file (if this file doesn't exist, `nfsd` reads */etc/exports* (p. 721) instead) to determine which mountpoints to service. Changes made to this file don't take effect until you either restart `nfsd` or you send `nfsd` a `SIGHUP` signal:

```
slay -s SIGHUP nfsd
```

There's no direct check for `root` on the mount; `nfsd` checks only that requests come in on a privileged port, which implies `root` access.

The `nfsd` daemon supports a maximum of 15 nested directory levels.

The `nfsd` command doesn't tolerate any parsing errors while reading the exports file. If an error is detected, `nfsd` terminates. To keep downtime to a minimum if you modify the exports file, we recommend that you either:

- Start another `nfsd`. If no parsing errors are detected, the second `nfsd` reports a bind failure, and exits — this indicates the exports file was parsed correctly.

  or:

- Start a second instance of `nfsd`, specifying the -P option.

Security Issues

NFS is a very unsecure protocol. Although `nfsd` checks each request's origin against restrictions specified in the exports file, this helps only in an “honest” network. It's not difficult to spoof NFS requests.

Configuring Caches

Fine-tuning `nfsd` caches may result in less memory usage and improved performance, but these qualities are usually exclusive. Before modifying the default behavior of `nfsd`, it's important to know what its clients will demand from it. Also note that these caches are shared across all mountpoints.

Copyright © 2014, QNX Software Systems Limited

/etc/nfsstart

Start NFS server daemons

Name:

`/etc/nfsstart`

Description:

The `nfsstart` file is a shell script that you can use to start TCP/IP daemons specific to an NFS server.

There isn't a default version of this file; you can create your own if you need it.

Here's a typical sample file that you should optimize on a per-system basis:

```
#!/bin/sh

  /usr/sbin/nfsd
  # NOTE:  to service non-Unix clients, uncomment the next line
  #        (may also need to add the -norsvd option in the
  #         exports file)
  # /usr/sbin/pcnfsd
```

Copyright © 2014, QNX Software Systems Limited

nice

Run a program at an altered priority (POSIX)

Syntax:

```
nice [-npriioritylevels] command [arguments]...
```

Deprecated:

```
nice [-prioritylevels] command [arguments]...
```

Runs on:

QNX Neutrino

Options:

-prioritylevels

Deprecated. This is the historical method of specifying the amount to adjust the current priority by. Specifying -1 lowers the priority by one, while specifying --1 boosts the priority by one.

-n ***prioritylevels***

Specify the amount to adjust the current priority by when running the command. The command is run at a priority level of the current priority minus *prioritylevels*.

command **[*arguments*]...**

The command to run at the altered priority.

Description:

The `nice` utility invokes the specified command with a modified priority, usually making the command behave "more nicely" towards competing processes.

If no *prioritylevels* option is specified, the program is invoked at a priority that's one level lower than the parent's current priority (i.e. it is invoked with a "nice increment" of 1).

If a *prioritylevels* option is specified, it's subtracted from the parent's current priority and the program is invoked at the resultant priority. If the resulting priority isn't a valid priority, `nice` writes a diagnostic message to the standard error and exits with a status of 1.

Copyright © 2014, QNX Software Systems Limited

If you enter:	nice:
A positive value (e.g. -n2 or -n+2)	*Lowers* the priority of the program, making it “nice”
A negative value (e.g. -n-2)	*Raises* the priority of the program, making it “mean”

You can adjust the priority as follows:

If you're:	You can change to any priority:
A non-`root` user	From 1 to 63
`root`	From 1 to 255

You can change the range of privileged priorities with the -P option for *procnto* (p. 1586).

Examples:

Run *make* (p. 1166) at one priority lower than the parent's priority (be nice):

```
nice make application
```

Run `make` at two priorities lower than the parent's priority (be nicer):

```
nice -n2 make application
```

Run `make` at two priorities *higher* than the parent's priority (be mean):

```
nice -n-2 make application
```

Exit status:

If the operation is successful, the exit status of the invoked command is returned. If an error occurs, the exit status is as follows:

1

Invalid command-line parameters were given or the user requested an invalid priority.

126

The *command* specified didn't exist.

127

The *command* couldn't be started for some other reason.

Copyright © 2014, QNX Software Systems Limited

Caveats:

In contrast to other operating systems, the QNX Neutrino interpretation of the nice value substantially affects the priority of the process. Rather than representing a *fraction* of a priority, the granularity of the nice value in QNX Neutrino is of a "whole" priority level. For example, where the following has a marginal effect on the execution of `myprog` on some operating systems:

```
nice -n5 myprog
```

on QNX Neutrino it lowers the priority of `myprog` by five full priority levels, and could have a significant effect on `myprog`'s execution time.

Copyright © 2014, QNX Software Systems Limited

nicinfo

Display information about a network interface controller

If you aren't `root`, specify the full path: `/usr/sbin/nicinfo`.

Syntax:

```
nicinfo [-c|g|s] [-rv] [interface|device...]
```

Runs on:

QNX Neutrino

Options:

-c

Display information about the configuration only.

-g

Display general statistics only.

-r

The specified devices are relative to `/dev/io-net/`. This option is for compatibility with older `io-net` drivers.

-s

Display statistics only.

-v

Be verbose.

device

The device (e.g. `/dev/io-net/en0`) to display information about. This is for compatibility with older `io-net` drivers; native `io-pkt` drivers don't create entries in `/dev`.

interface

The network interface (e.g. `fxp0`) to display information about. If you don't specify an interface, `nicinfo` displays information about all interfaces.

Copyright © 2014, QNX Software Systems Limited

You should specify at most one of the -c, -g, and -s options.

Description:

This utility displays information about the given network interface connection, or all interfaces if you don't specify one. The information includes the number of packets transmitted and received, collisions, and other errors.

- The output from `nicinfo` depends on what the driver supports; not all fields are included for all drivers. However, the output always includes information about the bytes and packets that were transmitted and received.
- NetBSD drivers don't support `nicinfo`.

Examples:

```
nicinfo en0

SMC9432 EPIC Ethernet Controller

  Physical Node ID ........................... 00E029 3820EE
  Current Physical Node ID ................... 00E029 3820EE
  Current Operation Rate ..................... 100.00 Mb/s half-duplex
  Active Interface Type ...................... MII
    Active PHY address ....................... 3
  Maximum Transmittable data Unit ............ 1514
  Maximum Receivable data Unit ............... 1514
  Hardware Interrupt ......................... 0x9
  I/O Aperture ............................... 0xb000 - 0xb0ff
  Memory Aperture ............................ 0xf1800000 - 0xf1800fff
  Promiscuous Mode ........................... Off
  Multicast Support .......................... Enabled

  Packets Transmitted OK ..................... 1096267
  Bytes Transmitted OK ....................... 1096353794
  Memory Allocation Failures on Transmit ..... 0

  Packets Received OK ........................ 1132010
  Bytes Received OK .......................... 1201171760
  Memory Allocation Failures on Receive ...... 0

  Single Collisions on Transmit .............. 744536
  Deferred Transmits ......................... 262485
  Late Collision on Transmit errors .......... 0
  Transmits aborted (excessive collisions) ... 0
  Transmit Underruns ......................... 0
  No Carrier on Transmit ..................... 0
  Receive Alignment errors ................... 0
  Received packets with CRC errors ........... 0
  Packets Dropped on receive ................. 0
```

Get statistics on legacy `io-net` drivers running with the shim:

```
nicinfo -r en0
```

Another way to do the same thing:

```
nicinfo /dev/io-net/en0
```

Get statistics on a particular native driver:

```
nicinfo wm0
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

Successful completion.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

nm

List symbols from object files (POSIX)

Syntax:

nm_variant [*options*] [*objfile*…]

where *nm_variant* depends on the target platform, as follows:

Target platform	*nm_variant*
ARMv7	`ntoarmv7-nm`
x86	`ntox86-nm`

Runs on:

Linux, Microsoft Windows

Description:

The `nm` utility lists the symbols from the specified object files. If no object files are listed as arguments, `nm` assumes `a.out`. For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

nohup

Invoke a command immune to hangups (POSIX)

Syntax:

nohup *command* [*argument*...]

Runs on:

QNX Neutrino

Options:

argument

The argument(s) for the command to be invoked.

Description:

The nohup utility invokes the specified command with arguments supplied as the *argument* operands. When the command is invoked, the SIGHUP signal is set to be ignored; thus, the command is made immune to hangups.

If the standard output is a terminal, all output written by the specified command to its standard output is appended to the nohup.out file in the current directory. If this nohup.out file can't be created, or if it can't be opened for appending, the output is appended to the end of the nohup.out file in the directory specified by the ***HOME*** environment variable. If neither nohup.out file can be created or opened, the specified command isn't invoked.

If the standard error output is a terminal, all output written by the specified command to its standard error output is redirected to the same file descriptor as the standard output.

It's often desirable to apply nohup to pipelines or lists of commands. You can do this by placing pipelines and command lists in a single file; this file can then be executed as a command, and the “nohup” applies to everything in the file.

Alternatively, the following can be used to apply nohup to a complex command:

```
nohup sh -c 'complex-command-line'
```

Similarly, you can use the following commands to apply nohup to a shell function:

```
export -f func
nohup sh -c 'command line invoking func'
```

Copyright © 2014, QNX Software Systems Limited

If you want the SIGQUIT signal to also be ignored, you can run nohup in the background:

```
nohup command &
```

Files:

./nohup.out

If possible, this file will be created and the output of the command will be written to it.

***$HOME*/nohup.out**

If it was not possible to open ./nohup.out, nohup will attempt to create (if necessary) and append the output of the command to this file. If neither file can be opened, nohup will not run the command.

Environment variables:

HOME

If the output file nohup.out cannot be created in the current directory, nohup uses the directory named by this variable to create the file.

PATH

Contains the search path used to locate the command to be invoked.

Exit status:

The exit status of nohup cannot be relied upon.

License:

This utility is based on copyright software of The Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Copyright © 2014, QNX Software Systems Limited

nslookup

Query Internet name servers interactively

Syntax:

```
nslookup [-options] [host-to-find | -[server]]
```

Runs on:

QNX Neutrino

Options:

options

Any option from the *set* (p. ?) command.

host-to-find

The host to look up.

server

The server to use for the lookup.

Description:

The `nslookup` utility lets you query Internet domain name servers. The utility has two modes: interactive and noninteractive. In interactive mode, you can query name servers for information about various hosts and domains or print a list of hosts in a domain. In noninteractive mode, `nslookup` just prints the name and requested information for a host or domain.

The utility enters interactive mode when:

- no arguments are given (the default name server is used)
- the first argument is a hyphen (`-`) and the second argument is the hostname or Internet address of a name server.

The utility enters noninteractive mode when the first argument is the name or Internet address of the host to be looked up. The optional second argument specifies the host name or address of a name server.

You can specify the options listed under the `set` command below in the `.nslookuprc` file in your home directory; list each one on its own line.

Copyright © 2014, QNX Software Systems Limited

You can specify interactive commands on the command line if they precede the arguments and you prefix them with a hyphen. For example, to change the default query type to host information and the initial timeout to 10 seconds, type:

```
nslookup -querytype=HINFO -timeout=10
```

Interactive commands

To interrupt a command at any time, press **Ctrl**–**C**. To exit, press **Ctrl**–**D** (EOF) or type `exit`. To treat a builtin command as a host name, place an escape character (\) before the command.

The length of the command line must be less than 256 characters.

Any unrecognized command is interpreted as a host name.

***host* [*server*]**

Look up information for *host* using the current default server or using *server*, if specified.

If *host* is an Internet address and the query type is A or PTR (see the `set querytype` command), the name of the host is returned. If *host* is a name and doesn't have a trailing period, the default domain name is appended to the name. (This behavior depends on the state of the `set` options `domain`, `srchlist`, `defname`, and `search`).

To look up a host *not* in the current domain, append a period to the name.

server *domain* or lserver *domain*

Change the default server to *domain*. The lserver form uses the initial server to look up information about *domain* while server uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root

Change the default server to the server for the root of the domain namespace. Because the host `ns.nic.ddn.mil` is currently used, this command is a synonym for `lserver ns.nic.ddn.mil`. You can change the name of the root server with the `set root` command.

finger [*name*] [> *filename*] or finger [*name*] [>> *filename*]

Connect with the finger server on the current host. The current host is defined when a previous lookup for a host completed successfully and returned

Copyright © 2014, QNX Software Systems Limited

address information (see the `set querytype`=A command). The *name* argument is optional. Note that you can use > and >> in the usual manner.

ls [*option*] *domain* [> *filename*] or ls [*option*] *domain* [>> *filename*]

List the information available for *domain*, optionally creating or appending to *filename*. If no option is given, the output contains hostnames and their Internet addresses. The *option* argument can be one of the following:

-t *querytype*

List all records of the specified type (see querytype below).

-a

List aliases of hosts in the domain. Synonym for -t CNAME.

-d

List all records for the domain. Synonym for -t ANY.

-h

List CPU and OS information for the domain. Synonym for -t HINFO.

-s

List well-known services of hosts in the domain. Synonym for -t WKS. When output is directed to a file, hash marks are printed for every 50 records received from the server.

view *filename*

Sort and list the output of previous `ls` commands with `more`.

help or ?

Print a brief summary of commands.

exit

Exit the utility.

All of the keywords on the following pages belong to the set command.

set *keyword*[=*value*]

Change state information that affects the lookups. Valid keywords are:

Copyright © 2014, QNX Software Systems Limited

all

Print the current values of `set`'s most frequently used options as well as information about the current default server and host.

class=*value*

Change the query class to one of:

IN

Internet class.

CHAOS

Chaos class.

HESIOD

MIT Athena Hesiod class.

ANY

Query for any of the above.

The class specifies the protocol group of the information (default = `IN`, abbreviation = `cl`).

[no]debug

Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer (default = `nodebug`, abbreviation = `[no]deb`).

[no]d2

Turn exhaustive debugging mode on. All fields of every packet are printed (default = `nod2`).

domain=*name*

Change the default domain name to *name*. The default domain name is appended to a lookup request depending on the state of the defname and search options (see below).

The domain search list contains the parents of the default domain if the domain has at least two components in its name. For example, if the default domain is `CC.Berkeley.EDU`, the search list is `CC.Berkeley.EDU` and `Berkeley.EDU` (default is the value from `hostname`, */etc/nsswitch.conf* (p. 1369) file, or the ***LOCALDOMAIN*** environment variable; the abbreviation is `do`).

Copyright © 2014, QNX Software Systems Limited

To specify a different list, use `set srchlist` (see below); to display the list, use `set all`.

srchlist=*name1*/*name2*/...

Change the default domain name to *name1* and the domain search list to *name1*, *name2*, etc. (default = value based on `hostname`, `/etc/nsswitch.conf` file, or the ***LOCALDOMAIN*** environment variable; abbreviation = `srchl`).

You can specify a maximum of six names separated by slashes (`/`). For example, this command:

```
set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU
```

sets the domain to `lcs.MIT.EDU` and the search list to the three names.

This command overrides the default domain name and search list of the `set domain` command. Use the `set all` command to display the list.

[no]defname

Append the default domain name to a single-component lookup request, i.e. one that doesn't contain a period (default = `defname`, abbreviation = `[no]def`).

[no]search

If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received (default = `search`, abbreviation = `[no]sea`).

port=*value*

Change the default TCP/UDP name server port to *value* (default = `53`, abbreviation = `po`).

querytype=*value* or type=*value*

Change the type of information query to one of:

A

The host's Internet address.

CNAME

Copyright © 2014, QNX Software Systems Limited

The canonical name for an alias.

HINFO

The host CPU and OS type.

MINFO

The mailbox, or mail-list information.

MX

The mail exchanger.

NS

The name server for the named zone.

PTR

The hostname if the query is an Internet address; otherwise the pointer to other information.

SOA

The domain's "start-of-authority" information.

TXT

The text information.

UINFO

The user information.

WKS

The supported well-known services. Other types (`ANY`, `AXFR`, `MB`, `MD`, `MF`, `NULL`) are described in the *RFC 1035* document.

Default = `A`; abbreviations = `q, ty`.

[no]recurse

Tell the name server to query other servers if it doesn't have the information (default = `recurse`, abbreviation = `[no]rec`).

retry=*number*

Set the number of retries to *number*. When a reply to a request isn't received within a certain amount of time (changed with `set timeout`), the timeout period is doubled and the request is

Copyright © 2014, QNX Software Systems Limited

sent again. The *number* argument controls how many times a request is resent before `nslookup` gives up (default = 4, abbreviation = `ret`).

root=host

Change the name of the root server to *host*. This affects the `root` command (default = `ns.nic.ddn.mil.`, abbreviation = `ro`).

timeout=*number*

Change the initial timeout interval for waiting for a reply to the specified number of seconds. Each retry doubles the timeout period (default = 5, abbreviation = `ti`).

[no]vc

Always use a virtual circuit when sending requests to the server (default = `novc`, abbreviation = `[no]v`).

[no]ignoretc

Ignore packet truncation errors (default = `noignoretc`, abbreviation = `[no]ig`).

Diagnostics:

If the lookup request wasn't successful, one of the following error messages may be printed:

`Timed out`

The server didn't respond to a request after a certain amount of time (changed with `set timeout=`*value*) and a certain number of retries (changed with `set retry=`*value*).

`No response from server`

No name server is running on the server machine.

`No records`

The server doesn't have resource records of the current query type for the host, although the hostname is valid. The query type is specified with the `set querytype` command.

`Nonexistent domain`

The hostname or domain name doesn't exist.

`Connection refused — Network is unreachable`

Copyright © 2014, QNX Software Systems Limited

The connection to the name or finger server couldn't be made at the current time. This error commonly occurs with `ls` and `finger` requests.

`Server failure`

The name server found an internal inconsistency in its database and couldn't return a valid answer.

`Refused`

The name server refused to service the request.

`Format error`

The name server found that the request packet wasn't in the proper format. This may indicate an error in `nslookup`.

Based on:

- *Domain Names — Concepts and Facilities (RFC 1034)*
- *Domain Names — Implementation and Specification (RFC 1035)*

Files:

`/etc/nsswitch.conf` **(p. 1369)**

Name-service switch configuration file.

`$HOME/.nslookuprc`

User's initial options.

`/etc/nslookup.help`

Summary of commands.

The `nslookup` utility requires the `libsocket.so` shared library.

Environment variables:

`HOSTALIASES`

Contains host aliases.

`LOCALDOMAIN`

Overrides default domain.

Contributing author:

Andrew Cherenson

Copyright © 2014, QNX Software Systems Limited

/etc/nsswitch.conf

Name-service switch configuration file

Name:

/etc/nsswitch.conf

Description:

The nsswitch.conf file specifies how the *nsdispatch()* (name-service switch dispatcher) routines in the socket library should operate.

The configuration file controls how a process looks up various databases containing information regarding hosts and networks. Each database comes from a source (such as local files, and DNS), and the order to look up the sources is specified in nsswitch.conf.

Each entry in nsswitch.conf consists of a database name, and a space-separated list of sources. Each source can have an optional trailing criterion that determines whether the next listed source is used, or the search terminates at the current source. Each criterion consists of one or more status codes, and actions to take if that status code occurs.

Sources

The following sources are implemented:

files

Local files, such as */etc/hosts* (p. 946).

dns

Internet Domain Name System. The hosts and networks databases use IN class entries.

Databases

The following databases are used by the following C library functions:

Database	Used by
hosts	*getaddrinfo()*, *gethostbyaddr()*, *gethostbyname()*, and *getnameinfo()*
networks	*getnetbyaddr()*, *getnetbyname()*

Status codes

The following status codes are available:

Copyright © 2014, QNX Software Systems Limited

success

The requested entry was found.

notfound

The entry isn't present at this source.

tryagain

The source is busy, and may respond to retries.

unavail

The source isn't responding, or the entry is corrupt.

Actions

For each of the status codes, one of two actions is possible:

continue

Try the next source.

return

Return with the current result.

Format of the file

A BNF description of the syntax of `nsswitch.conf` is:

```
<entry>        ::= <database> ":" [<source> [<criteria>]]*
<criteria>     ::= "[" <criterion>+ "]"
<criterion>    ::= <status> "=" <action>
<status>       ::= "success" | "notfound" | "unavail" | "tryagain"
<action>       ::= "return" | "continue"
```

Each entry starts on a new line in the file. A number sign (#) delimits a comment to the end of the line. Blank lines are ignored. A backslash (\) at the end of a line escapes the newline, and causes the next line to be a continuation of the current line. All entries are case-insensitive.

The default criteria is to return on `success`, and continue on anything else (i.e. `[success=return notfound=continue unavail=continue tryagain=continue]`).

Historically, many of the databases had enumeration functions, often of the form *getXXXent()*. These made sense when the databases were in local files, but don't make sense or have lesser relevance when there are possibly multiple sources, each of an unknown size. The interfaces are still provided for compatibility, but the source may not be able to provide complete entries, or duplicate entries may be retrieved if multiple sources that contain similar information are specified.

Copyright © 2014, QNX Software Systems Limited

If, for any reason, `nsswitch.conf` doesn't exist, or it has missing or corrupt entries, *nsdispatch()* will default to an entry of `files` for the `network` database, and `files dns` for the `hosts` database.

Luke Mewburn wrote this freely distributable name-service switch implementation, using ideas from the ULTRIX `svc.conf` and Solaris `nsswitch.conf` manual pages.

Examples:

To look up hosts in `/etc/hosts` and then from the DNS, use:

```
hosts:     files dns
```

Copyright © 2014, QNX Software Systems Limited

nsupdate

Dynamic DNS update utility

Syntax:

```
nsupdate [-d] [[-y [hmac:]keyname:secret] | [-k keyfile]]
         [-t timeout] [-u udptimeout] [-r udpretries]
         [-R randomdev] [-v] [filename]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?nsupdate++NetBSD-5.0* in the NetBSD documentation.

Description:

The `nsupdate` utility is used to submit Dynamic DNS Update requests as defined in RFC2136 to a name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?nsupdate++NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

ntpd

Network Time Protocol (NTP) daemon

Syntax:

```
ntpd [-46aAbdgLmnNpqx] [-c conffile] [-D level]
     [-f driftfile] [-i jaildir]
     [-k keyfile] [-l logfile] [-p pidfile]
     [-P priority] [-r broadcastdelay]
     [-s statsdir] [-t key]
     [-u user[:group] ] [-v variable] [-V variable]
```

Runs on:

QNX Neutrino

Options:

-4

Force DNS resolution of hosts to the IP4 namespace.

-6

Force DNS resolution of hosts to the IP6 namespace.

-a

Require cryptographic authentication for broadcast client, multicast client, and symmetric passive associations. This is the default.

-A

Don't require cryptographic authentication for broadcast client, multicast client, and symmetric passive associations.

-b

Enable the client to synchronize to broadcast servers.

-c *conffile*

Specify the name and path of the configuration file; the default is `/etc/ntp.conf`. For information about this file, see its entry in the *FreeBSD documentation*.

-d

Use debugging mode. You can specify this option more than once, with each occurrence producing a greater amount of detail

Copyright © 2014, QNX Software Systems Limited

-D ***level***

Specify the debugging level directly.

-f ***driftfile***

Specify the name and path of the frequency file; the default is as specified by `driftfile` *driftfile* configuration command, if present in the `ntp.conf` file.

-g

Set the panic threshold to any value without any restriction. If the offset exceeds the default value (1000 s) of this panic threshold, `ntpd` exits with a message to the system log. You can use this option with the -q and -x options. See the `tinker` command for other options.

-i ***jaildir***

Change the server's root directory to be *jaildir*. This option also implies that the server attempts to drop `root` privileges at startup (otherwise, changing the root directory gives little additional security), and it's available only if the OS supports running the server without full `root` privileges. You may need to also specify the -u option.

-k ***keyfile***

Specify the name and path of the symmetric key file. This is the same operation as the `keys` *keyfile* configuration command. For information about this file, see its entry in the *FreeBSD documentation*.

-l ***logfile***

Specify the name and path of the log file. The default is the system log file. This is the same operation as the `logfile` *logfile* configuration command.

-L

Don't listen to virtual IPs. The default is to listen.

-m

Enable the client to synchronize the multicast servers at the IPv4 multicast group address 224.0.1.1.

-n

Don't fork.

-N

Run the `ntpd` utility at the highest priority.

Copyright © 2014, QNX Software Systems Limited

-p ***pidfile***

Specify the name and path of the file used to record the `ntpd` process ID. This is the same operation as the `pidfile` *pidfile* configuration command.

-P ***priority***

Run the `ntpd` utility at the given priority.

-q

Exit the `ntpd` utility just after the first time the clock is set. You can use the -g and -x options with this option.

-r ***broadcastdelay***

Specify the default propagation delay from the broadcast/multicast server to this client. This is necessary if the protocol can't automatically compute the delay.

-s ***statsdir***

Specify the directory path for files created by the statistics facility. This is the same operation as the `statsdir` *statsdir* configuration command.

-t ***key***

Add a key number to the trusted key-list. This option can occur more than once.

-u ***user*[:*group*]**

Specify a user, and optionally a group, to switch to.

-v ***variable*** **or -V** ***variable***

Add the given system variable.

-x

Set the step threshold to 600 s. The default value is 128 ms. If the offset is less than (or above) the step threshold, the time is adjusted (or stepped up). You can use this option with the -g and -q options. Please also see the `tinker` command.

Description:

Use the `ntpd` utility to set and maintain the system time of day in synchronism with the Internet standard time servers. This utility is an operating system daemon that conforms to the NTP (Network Time Protocol) version 4 specification. It also retains compatibility with version 3, as defined by *RFC 1305*, and version 1 and 2, as defined

Copyright © 2014, QNX Software Systems Limited

by *RFC 1059* and *RFC 1119*. Using the 64-bit floating point arithmetic, the `ntpd` utility attempts to preserve the ultimate precision, which is about 232 picoseconds. While the ultimate precision isn't achievable with today's workstations and networks, it's required for future gigahertz CPU clocks and gigabit LANs.

The `ntpd` utility normally reads the `ntp.conf` file for its configuration. For information about this file, see its entry in the *FreeBSD documentation*.

Basic operation of the `ntpd` utility

The `ntpd` utility exchanges messages with one or more configured hosts at designated poll intervals. This is done to groom data and set the clock. Although the default initial poll interval is 64 seconds, the poll interval for each server is delayed by an interval randomized over few seconds, in order to protect the network from bursts.

In order to maintain time during periods when the power is off, today's hardware incorporates a time-of-year (TOY) chip. When the machine is booted, the chip is used to initialize the operating system time. After the machine has synchronized to a NTP server, the operating system time gets synchronized and corrected from time to time.

This doesn't set the hardware clock; you can use the *`rtc`* (p. 1714) utility to set the time on the chip.

Under ordinary conditions, `ntpd` adjusts the clock in small steps so that the time scale is continuous. Under conditions of extreme network congestion, the `ntpd` algorithms discard sample offsets exceeding 128 ms, because:

- the roundtrip delay jitter can exceed three seconds
- the synchronization distance, which is equal to one-half the roundtrip delay plus error budget terms, can become very large.

When there is no TOY chip, and the client time is more than 1000 seconds from the server time, you should intervene and set the clock by hand — causing the `ntpd` utility to exit with a panic message to the system log.

Frequency discipline

The behavior of the `ntpd` utility at startup depends on the existence of `ntp.drift` frequency file. This file contains the latest estimate of the clock frequency error. If this file doesn't exist, the `ntpd` utility enters a special mode designed to quickly adapt to system clock oscillator time and frequency error. After approximately 15 minutes, when the time and frequency are set to nominal values, the `ntpd` utility starts tracking the time and frequency relative to the server. After one hour, the current frequency offset is written to `ntp.drift` frequency file. Since the `ntpd` frequency is initialized from this file, the `ntpd` utility enters into normal mode immediately during startup. Subsequently, the current frequency offset is written to the `ntp.drift` frequency file at hourly intervals.

Copyright © 2014, QNX Software Systems Limited

Operating modes

The ntpd utility operates continuously while monitoring for small changes in frequency, trimming the clock for ultimate precision. It operates in one of several modes:

- symmetric active/passive
- client/server
- broadcast/multicast
- manycast.

The ntpd utility also operates in one-time mode where the time is set from an external server, and the frequency is set from a previously recorded frequency file. When a client operates in broadcast/multicast or manycast mode, it first discovers the remote server, and then configures itself automatically — after computing the server-client propagation delay correction factors. You can now deploy a fleet of workstations without specifying configuration details that are specific to your local environment.

When the ntpd utility runs in continuous mode, each of the external servers is polled at intervals determined by an intricate state machine. The state machine determines the poll interval using a heuristic algorithm, after measuring the incidental roundtrip delay jitter and oscillator frequency wander. In most operating environments, the state machine starts with intervals of 64 seconds that increase to 1024 seconds, in steps. A small amount of random variation is also introduced, in order to avoid bunching at the servers.

In some cases, it may not be practical for ntpd to run continuously. A common workaround has been to run the ntpdate utility from a cron job at designated times. This utility, however, doesn't have the signal processing, error checking, and mitigation algorithms of the ntpd utility. The -q option is intended for this purpose.

In QNX Neutrino, a useful feature is available to discipline the clock frequency. First, the ntpd utility is run in continuous mode with selected servers in order to measure and record the intrinsic clock frequency offset. This may take hours (for the frequency and offset) to settle down. Then, the ntpd utility is stopped and run in one-time mode as required. At each startup, ntpd reads the frequency from the file and initializes the kernel frequency.

Poll interval control

The state machine for ntpd maintains synchronization consistent with the observed jitter and wander. You have several ways to tailor the operation of the ntpd utility, for example, either by reducing or increasing the poll interval. You must be careful, however, to consider the consequences of changing the adjustment range for poll intervals.

The default minimum for poll interval is 64 seconds, and the default maximum is 1,024 seconds.

Copyright © 2014, QNX Software Systems Limited

You can change the default minimum with the `tinker` *minpoll* command to a value not less than 16 seconds. This value is used for all configured associations, unless overridden by the *minpoll* option on the configuration command. Most of the device drivers don't operate properly if the poll interval is less than 64 seconds.

In some cases involving dialup or toll services, you may also increase the minimum interval to a few tens of minutes and the maximum interval to a day or so. Under normal operating conditions, once the clock discipline loop has stabilized, the interval is increased in steps from the minimum to the maximum value. This assumes, however, that the intrinsic clock frequency error is small enough for the discipline loop to correct it. The capture range of the loop is 500 PPM at an interval of 64 seconds, decreasing by a factor of two for each doubling of interval. At a minimum of 1,024 seconds, for example, the capture range is only 31 PPM. If the intrinsic error is greater than this, the `ntp.drift` drift file is specially tailored to reduce the residual error below this limit. Once this is done, the drift file is automatically updated once every hour and is available to initialize the frequency on subsequent daemon restarts.

The `huff-n'-puff` filter

Timekeeping quality may seriously degrade in certain scenarios, for example, when a considerable amount of data is to be downloaded or uploaded over telephone modems. This occurs because differential delays during transmission on two directions can be quite large. Sometimes, time errors even exceed the step threshold, resulting in step correction during and after the data transfer.

Use the `huff-n'-puff` filter to correct the time offset in these cases. It uses the knowledge of the propagation delay when no other traffic is present, i.e. during other than work hours. The filter maintains a shift register that remembers the minimum delay over the most recent interval that is measured in hours. Under conditions of severe delay, the filter corrects the apparent offset using the sign of the offset, and the difference between the apparent delay and minimum delay. The name of the filter reflects the negative (`huff`) and positive (`puff`) correction, which depends on the sign of the offset. The filter is activated by the `tinker` command and `huffpuff` keyword.

Based on:

RFC 1059, *RFC 1119*, *RFC 1305*

Copyright © 2014, QNX Software Systems Limited

ntpdate

Set the local time and date by polling NTP servers

Syntax:

```
ntpdate [-46bBdqsuv] [-a key] [-e authdelay]
        [-k keyfile] [-M i=your_hostname]
        [-o version]  [-p samples]
        [-t timeout] server [...]
```

Runs on:

QNX Neutrino

Options:

-4

Force DNS resolution of hosts to the IP4 namespace.

-6

Force DNS resolution of hosts to the IP6 namespace.

-a *key*

Enable the authentication function, where *key* is the key identifier. Both keys and the key identifiers must match the client and server key files. The default is to disable the authentication function.

-B

Force the time to always be adjusted using *ClockAdjust()*, even if the measured offset is larger than ±128 milliseconds. The default is to set the time using *settimeofday()*. In case the offset is much greater than ±128 milliseconds, a longer time (in hours) may be needed to adjust the clock to the correct value. During this time, the host is not used to synchronize clients.

-b

Force the time to be stepped using the *settimeofday()* function call. You should use this option when calling from a startup file at boot time.

-d

Enable the debugging mode, after going through all the steps. Don't adjust the local clock. Print the useful information.

Copyright © 2014, QNX Software Systems Limited

-e ***authdelay***

Specify the processing delay *authdelay* (in seconds and fractions thereof) to perform an authentication function. This option improves timekeeping for slow CPUs. You may neglect it for most purposes.

-k ***keyfile***

Specify the path for the authentication key file as the string *keyfile*. The default is `/etc/ntp.keys`. For information about this file, see its entry in the *FreeBSD documentation*.

-M i=*your_hostname*

Enable your host to listen broadcast and multicast messages.

-o ***version***

Specify the NTP version, which can be 1, 2, 3, or 4. The default is 4. Use this option to use `ntpdate` with an older NTP version.

-p ***samples***

Specify the number of samples acquire from each server. The range is 1 to 8; the default is 4.

-q

Query the clock; don't set it.

-s

Divert logging output to the system `syslog` facility. This is designed primarily for the convenience of `cron` scripts.

-t ***timeout***

Specify the maximum time for a server response — in seconds and fractions thereof. The value is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

-u

Direct `ntpdate` to use an unprivileged port for outgoing packets. This is most useful when a firewall blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall. Note that the -d option always uses unprivileged ports.

-v

Be verbose. Log the `ntpdate` version identification string.

Copyright © 2014, QNX Software Systems Limited

Description:

Use the `ntpdate` utility to set the local time and date by polling NTP (Network Time Protocol) servers. The accuracy and reliability of this utility depends on the number of servers, the number of polls each time it is run, and the interval between runs.

Run the `ntpdate` utility as `root` on the local host.

At boot time, you may use a host startup script to run the `ntpdate` utility in order to set the clock. If necessary, you can also run this utility manually. Using the host script to set time initially is sometimes useful, e.g. set the time before you start the NTP daemon, `ntpd`. You can also run `ntpdate` from a `cron` script. The accuracy of the `ntpdate` utility is limited.

You use an -M option to support broadcast and multicast messages. Use -M i=*hostname* to join the multicast group. You are now able to listen for broadcast and multicast messages from an broadcast/multicast enabled server.

Use the `ntpdate` utility to adjust time in two ways:

- Use *settimeofday()* to step the time when the clock's in error by more than 0.5 seconds.
- Use *ClockAdjust()* to adjust the time when the clock's in error by less than 0.5 seconds.

The latter technique is less disruptive and more accurate when the error is small, and works quite well when `ntpdate` is run by `cron` every hour or two.

When the `ntpd` utility is running on the same host, the `ntpdate` utility doesn't set the date.

You may force the DNS resolution to the IPv4 (or IPv6) namespace, if you use a -4 (or -6) option that precedes a host name.

Caveats:

Since the time adjustment is actually 50% larger than the measured offset, it tends to keep a badly drifting clock more accurate. This is not good, however, for some kernel variables such as *tick* or *tickadj*.

Copyright © 2014, QNX Software Systems Limited

ntpdc

Query the NTP daemon

Syntax:

```
ntpdc [-46ilnps] [-c command] [host] [...]
```

Runs on:

QNX Neutrino

Options:

-4

Force DNS resolution of hosts to the IP4 namespace.

-6

Force DNS resolution of hosts to the IP6 namespace.

-c *command*

Execute the given command on the specified hosts. You can use multiple -c options. For more information about the commands, see below.

-i

Force the `ntpdc` utility to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

-l

Obtain a list of peers that are known to the servers. It is equivalent to -c `listpeers` option.

-n

Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.

-p

Print a list of the `peers` known to the server along with the summary of their state. This is equivalent to the -c `peers` option.

-s

Copyright © 2014, QNX Software Systems Limited

Print a list of the peers known to the server along with the summary of their state. It has slightly different format than the -p option. This is equivalent to -c `dmpeers` command.

Description:

Use the `ntpdc` utility to query the `ntpd` daemon about its current state, and to request changes in that state. You can run this utility is run either in interactive mode or in command mode. It provides extensive state and statistics information. At run time, all configuration options that are specified at startup using the `ntpd` utility's configuration file can also be specified using the `ntpdc` utility.

When you run the `ntpdc` utility by including one or more requests in the command line, each request is sent to the NTP (Network Time Protocol) servers running on each of the hosts. If no request option is given, the `ntpdc` utility attempts to read commands from the standard input and execute them on the NTP server running on the first host, as given on the command line. If no host is mentioned, it always defaults to `localhost`. The `ntpdc` utility prompts for commands if the standard input is a terminal device.

The `ntpdc` utility uses NTP mode 7 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network that permits it. However it is somewhat unreliable, especially over large distances in a network topology. The `ntpdc` utility makes no attempt to retransmit requests, and times out if the remote host's response isn't received within a suitable timeout time.

NTP behaves very similar to UDP (User Datagram Protocol).

You may force the DNS resolution to the IPv4 (or IPv6) namespace, if you use a -4 (or -6) option before a host name.

Interactive commands

The interactive format commands consist of a keyword followed by zero or more arguments. You can type only enough characters to uniquely identify the command. The output of a command is normally sent to the standard output, but you can send output of individual commands may be sent to a file by appending a <, followed by a file name, to the command line. A number of interactive format commands are executed entirely within the `ntpdc` utility:

? [*command_keyword*] or `help` [*command_keyword*]

Print a list of all the command keywords for the `ntpdc` utility. If you specify a command keyword, the function and the usage information about the command are printed.

Copyright © 2014, QNX Software Systems Limited

delay *milliseconds*

Specify a time interval. This is to be added to timestamps for requests that require authentication.

host *hostname*

Set the host to which future queries will be sent. The *hostname* may be either a host name or a numeric address.

hostnames [yes | no]

Print the host names in the information display when yes is specified. Print the numeric address when no is specified. The default is yes, unless modified using the command-line -n option.

keyid *keyid*

Set the key number to use to authenticate configuration requests. This must correspond to a key number that the server has been configured to.

passwd

Prompt for a password, which is not echoed, and is used to authenticate configuration requests. The password must correspond to the key configured for the NTP server for this purpose.

quit

Exit the ntpdc utility.

timeout *milliseconds*

Specify a timeout period for responses to server queries. The default is about 8000 milliseconds. Since the ntpdc utility retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

Control message commands

When you use the ntpdc utility to query, NTP mode 7 packets containing requests are sent to the server. These are read-only commands in that they make no modification of the server configuration state.

listpeers

Obtain and print a brief list of the peers for which the server is maintaining the state. These should include all configured peer associations as well as those peers whose stratum is such that they are considered by the server to be possible future synchronization candidates.

Copyright © 2014, QNX Software Systems Limited

peers

Obtain a list of peers for which the server is maintaining the state, along with a summary of that state. Summary information includes the address of the remote peer, the local interface address (0.0.0.0 if a local address has yet to be determined), the stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized), the polling interval, in seconds, the register in octal, and the current estimated delay, offset and dispersion of the peer, all in seconds.

The character in the left margin indicates the mode this peer entry is operating in.

This Character:	Indicates:
+	Symmetric active
-	Symmetric passive
=	Remote server is being polled in client mode
^	Server is broadcasting to this address
*	Server is currently synchronizing to this peer.

The contents of the host field may be one of:

- a host name
- an IP address
- a reference clock implementation name with its parameter or `REFCLK(implementation number, parameter)`.

If you've specified `no`, only IP-addresses are displayed.

dmpeers

Obtain peer summary list, identical to the output of the `peers` command, except for the character in the leftmost column. Characters appear only beside peers that were included in the final stage of the clock selection algorithm. A . indicates that this peer was cast off in the falseticker detection, while a + indicates that the peer made it through. A * denotes the peer the server is currently synchronizing with.

showpeer ***peer_address*** **[...]**

Show a detailed display of the current peer variables for one or more peers. Most of these values are described in the NTP version 2 specification.

Copyright © 2014, QNX Software Systems Limited

pstats *peer_address* **[...]**

Show per-peer statistic counters associated with the specified peers.

clockinfo *clock_peer_address* **[...]**

Obtain and print information concerning a peer clock. The values obtained provide information on the setting of fudge factors and other clock performance information.

kerninfo

Obtain and print kernel phase-lock loop operating parameters. This information is available only if the kernel has been specially modified for a precision timekeeping function.

loopinfo [oneline | multiline]

Print the values of selected loop filter variables. The loop filter is the part of NTP that deals with adjusting the local system clock. The *offset* is the last offset given to the loop filter by the packet-processing code. The *frequency* is the frequency error of the local clock in parts-per-million (ppm). The *time_const* controls the stiffness of the phase-lock loop and thus the speed at which it can adapt to oscillator drift. The *watchdog timer* value is the number of seconds which have elapsed since the last sample offset was given to the loop filter. The `oneline` and `multiline` options specify the format in which this information is to be printed, with `multiline` as the default.

sysinfo

Print a variety of system state variables, i.e., state related to the local server. All except the last four lines are described in the NTP Version 3 specification, *RFC 1305*.

The system flags show various system flags, some of which can be set and cleared by the `enable` and `disable` configuration commands, respectively. These are the `auth`, `bclient`, `monitor`, `pll`, `pps` and `stats` flags. See the *ntpd* (p. 1373) documentation for the meaning of these flags.

There are two additional flags which are read only, the `kernel_pll` and `kernel_pps`. These flags indicate the synchronization status when the precision time kernel modifications are in use. The `kernel_pll` indicates that the local clock is being disciplined by the kernel, while the `kernel_pps` indicates the kernel discipline is provided by the PPS signal.

The stability is the residual frequency error remaining after the system frequency correction is applied and is intended for maintenance and debugging. In QNX Neutrino, this value will initially decrease from as high

Copyright © 2014, QNX Software Systems Limited

as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the daemon, something may be wrong with the local clock, or the value of the kernel variable tick may be incorrect. The `broadcastdelay` shows the default broadcast delay, as set by the `broadcastdelay` configuration command. The `authdelay` shows the default authentication delay, as set by the authdelay configuration command.

`sysstats`

Print statistic counters maintained in the protocol module.

`memstats`

Print statistic counters related to memory-allocation code.

`iostats`

Print statistic counters maintained in the input-output module.

`timerstats`

Print statistic counters maintained in the timer/event queue support code.

`reslist`

Obtain and print the server's restriction list. This list is (usually) printed in sorted order and may help to understand how the restrictions are applied.

`monlist` **[*version*]**

Obtain and print traffic counts collected and maintained by the monitor facility. You don't normally have to specify the version number.

`clkbug` ***clock_peer_address*** **[...]**

Obtain debugging information for a reference clock driver. This information is provided only by some clock drivers and is mostly undecodable without a copy of the driver source in hand.

Runtime configuration requests

With the help of a configured NTP key, the server authenticates all requests. Authenticated requests always include a timestamp in the packet data, which is included in the computation of the authentication code. This timestamp is compared by the server to its receive timestamp. If they differ by more than a small amount, the request is rejected. The following commands all make authenticated requests:

`addpeer` ***peer_address*** **[*keyid*] [*version*] [`prefer`]**

Add a configured peer association at the given address and operating in symmetric active mode. Note that an existing association with the same peer

Copyright © 2014, QNX Software Systems Limited

may be deleted when this command is executed, or may simply be converted to conform to the new configuration, as appropriate.

If the optional *keyid* is a nonzero integer, all outgoing packets to the remote server will have an authentication field attached encrypted with this key. If the value is 0 (or not given) no authentication will be done. The version number can be 1, 2, 3, or 4, and defaults to 3. The `prefer` keyword indicates a preferred peer (and thus will be used primarily for clock synchronization if possible). The preferred peer also determines the validity of the PPS signal; if the preferred peer is suitable for synchronization, so is the PPS signal.

`addserver` ***peer_address*** **[*keyid*] [*version*] [*prefer*]**

Identical to the `addpeer` command, except that the operating mode is client.

`broadcast` ***peer_address*** **[*keyid*] [*version*] [*prefer*]**

Identical to the `addpeer` command, except that the operating mode is broadcast. In this case, a valid key identifier and key are required. The *peer_address* parameter can be the broadcast address of the local network or a multicast group address assigned to NTP. If a multicast address, a multicast-capable kernel is required.

`unconfig` ***peer_address*** **[...]**

This command causes the configured bit to be removed from the specified peer(s). In many cases, this causes the peer association to be deleted. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue in this fashion.

`fudge` ***peer_address*** **[*time1*] [*time2*] [*stratum*] [*refid*]**

This command provides a way to set certain data for a reference clock.

`enable` **[`auth` | `bclient` | `calibrate` | `kernel` | `monitor` | `ntp` | `pps` | `stats`]** **or** **`disable`** **[`auth` | `bclient` | `calibrate` | `kernel` | `monitor` | `ntp` | `pps` | `stats`]**

These commands operate in the same way as the `enable` and `disable` configuration file commands of *`ntpd`* (p. 1373).

`restrict` ***address mask flag*** **[*flag*]**

This command operates in the same way as the restrict configuration file commands of the `ntpd` utility.

`unrestrict` ***address mask flag*** **[*flag*]**

Copyright © 2014, QNX Software Systems Limited

Unrestrict the matching entry from the restriction list.

delrestrict ***address mask*** **[*ntpport*]**

Delete the matching entry from the restriction list.

readkeys

Purge the current set of authentication keys and obtain a new set by rereading the keys file (which must have been specified in the `ntpd` configuration file). This allows encryption keys to be changed without restarting the server.

trustedkey ***keyid*** **[...] or untrustedkey** ***keyid*** **[...]**

These commands operate in the same way as the `trustedkey` and `untrustedkey` configuration file commands of `ntpd`.

authinfo

Return information concerning the authentication module, including known keys and counts of encryptions and decryptions that have been done.

traps

Display the traps set in the server.

addtrap [*address* [*port*] [*interface*]

Set a trap for asynchronous messages.

clrtrap [*address* [*port*] [*interface*]

Clear a trap for asynchronous messages.

reset

Clear the statistics counters in various modules of the server.

Caveats:

The `ntpdc` utility is a crude hack. It is designed so that new (and temporary) features were easy to hack in, at great expense to the program's ease of use. Despite this, the program is occasionally useful.

Copyright © 2014, QNX Software Systems Limited

ntpq

Monitor the NTP daemon and determine its performance

Syntax:

```
ntpq [-46dinp] [-c command] [host] [...]
```

Runs on:

QNX Neutrino

Options:

-4

Force DNS resolution of hosts to the IP4 namespace.

-6

Force DNS resolution of hosts to the IP6 namespace.

-c *command*

Execute the given command on the specified hosts. You can use multiple -c options.

-d

Turn on the debugging mode.

-i

Force `ntpq` to operate in interactive mode. Prompts are written to the standard output and commands are read from the standard input.

-n

Print all host addresses in dotted-quad numeric format rather than converting them to the canonical host names.

-p

Print a list of peers known to the server, and a summary of their state. This is equivalent to the `peers` interactive command.

Description:

The `ntpq` utility monitors the `ntpd` daemon operations and determines its performance. It uses the standard NTP mode 6 control message formats defined in Appendix B of

Copyright © 2014, QNX Software Systems Limited

the NTPv3 specification *RFC 1305*. The same formats are also used for NTPv4 specification, which has more variables, and are discussed here.

You can run this utility either in interactive mode or in command mode. Command mode is controlled using command-line arguments. You can use both raw and pretty-printed options when assembling requests to read or write. You can also obtain and print a list of peers in a common format by sending multiple queries to the server.

When you run the `ntpq` utility by including one or more requests in the command line, each request is sent to the NTP servers running on each of the hosts. If no request option is given, `ntpq` attempts to read commands from the standard input and execute them on the NTP server running on the first host, as given on the command line. If no host is mentioned, it always defaults to `localhost`. The `ntpq` utility prompts for commands if the standard input is a terminal device.

The `ntpq` utility uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network that permits it. However it is somewhat unreliable, especially over large distances in a network topology. The `ntpq` utility makes only one attempt to retransmit requests, and times out if the remote host's response isn't received within a suitable timeout time.

NTP behaves very similar to UDP (User Datagram Protocol).

In contexts where a host name is expected, a -4 qualifier preceding the host name forces DNS resolution to the IPv4 namespace, while a -6 qualifier forces DNS resolution to the IPv6 namespace.

Specifying a command line option other than -i or -n causes the specified queries to be sent to the indicated host(s) immediately. Otherwise, `ntpq` attempts to read interactive format commands from the standard input.

Internal commands

The interactive format commands consist of a keyword followed by zero or more arguments. You can type only enough characters to uniquely identify the command. The output of a command is normally sent to the standard output, but you can send the output to a file by appending a <, followed by a file name, to the command line. A number of interactive format commands are executed entirely within the `ntpq` utility:

? [*command_keyword*] or `helpl` [*command_keyword*]

Print a list of all the command keywords for `ntpq` utility. If you specify a command keyword, the function followed by a command keyword, the function and the usage information about the command are printed.

`addvars` *variable_name* [= *value*] [...] or `rmvars` *variable_name* [...] or `clearvars`

Copyright © 2014, QNX Software Systems Limited

Allow variables and their optional values to be added to the list maintained internally by `ntpq`. If more than one variable is to be added, the list should be comma-separated and shouldn't contain white space. You can use the `rmvars` command to remove individual variables from the list. The `clearlist` command removes all variables from the list.

`cooked`

Cause the output from query commands to be "cooked," i.e. it reformats the values of the variables for useful purposes. The `ntpq` utility marks those variables that aren't decodable with a trailing `?`.

`debug more | less | off`

Turn debugging on and off.

`delay` ***milliseconds***

Specify a time interval. This is to be added to timestamps for requests that require authentication.

`host` ***hostname***

Set the host to which to send future queries. The *hostname* may be either a host name or a numeric address.

`hostnames [yes | no]`

Print the host names in the information display when `yes` is specified. Print the numeric address when `no` is specified. The default is `yes`, unless modified using the command-line -n option.

`keyid` ***keyid***

Specify the key number to use to authenticate configuration requests. This must correspond to a key number that the server has been configured to.

`ntpversion 1 | 2 | 3 | 4`

Set the NTP version number that the `ntpq` utility claims in packets. The default value is 3. Mode 6 control messages (and modes, for that matter) didn't exist in NTP version 1.

`passwd`

Prompt for a password, which isn't echoed, to use to authenticate configuration requests. The password must correspond to the key configured for NTP server for this purpose.

`quit`

Copyright © 2014, QNX Software Systems Limited

Exit the ntpq utility.

raw

Cause all output from query commands to be printed as received from the remote server. The only formatting/interpretation done on the data is to transform non-ASCII data into a printable (but barely understandable) form.

timeout ***millseconds***

Specify a timeout period for responses to server queries. The default is about 5000 milliseconds. Since the ntpq utility retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

Control message commands

A 16-bit (integer) association identifier is associated with an NTP server. When NTP control messages are sent, this association identifier is always included to identify peers. An association identifier of 0 has special meaning; it indicates that the variables are system variables, whose names are drawn from a separate name space.

Control message commands result in one or more NTP mode 6 messages, which are sent to the server, and data returned is always printed in some format. You will find that most commands send a single message and expect a single response. The current exceptions are the peers command, which sends a preprogrammed series of messages to obtain the required data, and the mreadlist and mreadvar commands, which iterate over a range of associations.

associations

Obtain and print a list of association identifiers and status for in-spec peers of the NTP servers you query. The list is printed in columns. The first column is an index, numbering the associations from 1 for internal use, the second column is the actual association identifier returned by the server, and the third column is the status word for the peer. The following columns contain data decoded from the status word.

The data returned by the associations command is cached internally in the ntpq utility. The index is useful when you deal with some servers that have association identifiers which are hard for humans to type. For any subsequent command that requires an association identifier as an argument, you can use the form and the index as an alternative.

clockvar [*assocID*] [*variable_name* [= *value* [...]] [...] or cv [*assocID*] [*variable_name* [= *value* [...]][...]

Copyright © 2014, QNX Software Systems Limited

Request to send a list of the server's clock variables. Servers that have radio clock or other external synchronization mechanism respond positively to this. If the association identifier is omitted or zero, the request for the variables of the system clock gets a positive response from all servers with a clock. If the server treats clocks as pseudo-peers, and has more than one clock connected, referencing the appropriate peer association identifier show the variables of a particular clock. Omitting the variable list causes the server to return a default variable display.

`lassociations`

Obtain and print a list of association identifiers and status of the `peers` for which the server is maintaining state. This command differs from the `associations` command only for servers that retain state for out-of-spec client associations. Such associations are normally omitted from the display when the `associations` command is used, but are included in the output of `lassociations`.

`lpassociations`

Print data for all associations, including out-of-spec client associations, from the internally cached list of associations. This command differs from `passociations`.

`lpeers`

Print a summary of all associations for which the server is maintaining the state. This produces a much longer list of peers.

`mreadlist` ***assocID assocID*** **or `mrl`** ***assocID assocID***

Behave like the `readlist` command, except the query is done for each of a range of (nonzero) association identifiers. This range is determined from the association list cached by the most recent `associations` command.

`mreadvar` ***assocID assocID*** **[***variable_name***[= *value*[...]** **or `mrv`** ***assocID assocID*** **[variable_name [= *value*[...]**

Behave like the `readvar` command, except the query is done for each of a range of (nonzero) association identifiers. This range is determined from the association list cached by the most recent `associations` command.

`opeers`

An old form of the `peers` command with the reference identifier replaced by the local interface address.

`passociations`

Copyright © 2014, QNX Software Systems Limited

Display association data concerning in-spec `peers` from the internally cached list of associations. This command performs identically to the `associations` command, except that it displays the internally stored data rather than making a new query.

`peers`

Obtain a current list of the peers, along with the state summary. Summary information includes the address of the remote peer, the reference identifier (0.0.0.0 if this is unknown), the stratum of the remote peer, and the type of the peer (local, unicast, multicast or broadcast). It also includes the polling interval in seconds, the register in octal, and the current estimated delay, offset, and dispersion of the peer, all in milliseconds. The character at the left margin of each line shows the synchronization status of the association and is a valuable diagnostic tool. The encoding and meaning of this character, called the tally code, is given later in this page.

`pstatus` ***assocID***

Send a read-status request to the server for the given association. Print the names and values of the `peer` variables that are returned. Note that the status word from the header is displayed preceding the variables, both in hexadecimal and in pidgin English.

`readlist` **[*assocID*] or `rl` [*assocID*]**

Request to return the variables in the internal variable list of the server. When the association identifier is omitted or 0, the variables are treated either as system variables, or peer variables. If the internal variable list is empty, a request is sent without data that induces the remote server to return a default display.

`readvar` ***assocID variable_name*** **[=*value*] [...] or `rv` *assocID* [*variable_name* [= *value*] [...]**

Request to return the values of the specified variables by sending a read variables request. If the association identifier is omitted or 0, the variables are treated either as system variables or peer variables that are returned of the corresponding peer. Omitting the variable list sends a request with no data, which induces the server to return a default display. The encoding and meaning of the variables derived from NTPv3 are given in *RFC 1305*; the encoding and meaning of the additional NTPv4 variables are given later in this page.

`writevar` ***assocID variable_name*** **[=*value*[...]**

Write the specified variables. Behave like the `readvar` request command.

Copyright © 2014, QNX Software Systems Limited

writelist [*assocID*]

Write the internal list of variables. Behave like the `readlist` request command.

Tally codes

The character in the left margin of the `peers` billboard, called the tally code, shows the fate of each association in the clock selection process. Following is a list of these characters, for which the `peer` is:

space reject

Discarded as unreachable, synchronized to this server (synch loop) or outrageous synchronization distance.

x falsetick

Discarded by the intersection algorithm as a falseticker.

. excess

Discarded as not among the first ten `peers` sorted by synchronization distance, and probably a poor candidate for further consideration.

- outlyer

Discarded by the clustering algorithm as an outlyer.

 # candidat

A survivor, and a candidate for the combining algorithm.

selected

A survivor, but not among the first six peers sorted by synchronization distance. If the association is ephemeral, it may be demobilized to conserve resources.

*** sys.peer**

Declared as the system `peer` and lends its variables to the system variables.

o pps.peer

Declared as the system `peer` and lends its variables to the system variables. The actual system synchronization is derived from a pulse-per-second (PPS) signal, either indirectly via the PPS reference clock driver or directly via the kernel interface.

System variables

Copyright © 2014, QNX Software Systems Limited

The *status*, *leap*, *stratum*, *precision*, *rootdelay*, *rootdispersion*, *refid*, *reftime*, *poll*, *offset*, and *frequency* variables are described in *RFC 1305* specification. Additional NTPv4 system variables include:

version

Software version and generation time.

processor

Processor and kernel identification string.

system

Operating system version and release identifier.

state

State of the clock discipline state machine. The values are described in the architecture briefing on the NTP project page linked from `www.ntp.org`.

peer

Internal integer used to identify the association currently designated as the system peer.

jitter

Estimated time error of the system clock measured as an exponential average of RMS time differences.

stability

Estimated frequency stability of the system clock measured as an exponential average of RMS frequency differences.

Additional system variables are displayed when the NTPv4 daemon is compiled with the OpenSSL software library.

flags

Current flags word bits and message digest algorithm identifier (NID) in hexadecimal format. The high-order 16 bits of the four-byte word contain the NID from the OpenSSL library, while the low-order bits are interpreted as follows:

`0x01`

Autokey enabled

`0x02`

Copyright © 2014, QNX Software Systems Limited

NIST leapseconds file loaded

0x10

PC identity scheme

0x20

IFF identity scheme

0x40

GQ identity scheme.

hostname

Host name as returned by *gethostname()*.

hostkey

NTP filestamp of the host key file.

cert

A list of certificates held by the host. Each entry includes the subject, issuer, flags and NTP filestamp in order. The bits are interpreted as follows, where the certificate:

0x01

Has been signed by the server.

0x02

Is trusted.

0x04

Is private.

0x08

Contains errors and shouldn't be trusted.

leapseconds

NTP filestamp of the NIST leapseconds file.

refresh

NTP timestamp when the host public cryptographic values are refreshed and signed.

Copyright © 2014, QNX Software Systems Limited

signature

Host digest/signature scheme name from the OpenSSL library.

tai

TAI-UTC offset in seconds obtained from the NIST leapseconds table.

Peer variables

The *status*, *srcadr*, *srcport*, *dstadr*, *dstport*, *leap*, *stratum*, *precision*, *rootdelay*, *rootdispersion*, *readh*, *hmode*, *pmode*, *hpoll*, *ppoll*, *offset*, *delay*, *dspersion*, and *reftime* variables are described in the *RFC 1305* specification, as are the timestamps *org*, *rec* and *xmt*. Additional NTPv4 peer variables include:

flash

Flash code for the most recent packet received. The encoding and meaning of these codes is given below.

jitter

Estimated time error of the `peer` clock measured as an exponential average of RMS time differences.

unreach

Value of the counter which records the number of poll intervals since the last valid packet was received.

When the NTPv4 daemon is compiled with the OpenSSL software library, additional peer variables are displayed, as follows:

flags

Current flag bits. This word is the server host status word with additional bits used by the Autokey state machine.

hostname

Server host name.

initkey

Initial key used by the key list generator in the Autokey protocol.

initsequence

Initial index used by the key list generator in the Autokey protocol.

signature

Copyright © 2014, QNX Software Systems Limited

Server message digest/signature scheme name from the OpenSSL software library.

timestamp

NTP timestamp when the last Autokey key list was generated and signed.

Flash codes

Use the `flash` code to debug. It is displayed in the `peer` variables list and shows the results of the original sanity checks defined in the NTP specification *RFC 1305* and additional ones added in NTPv4. There are 12 tests, designated as TEST1 through TEST12, that perform in a certain order designed to gain maximum diagnostic information while protecting against accidental or malicious errors. The *flash* variable is initialized to zero as each packet is received. If, after each set of tests, one or more bits are set, the packet is discarded. Use these tests for the following tasks:

TEST1 through TEST3

Check the packet timestamps from which the offset and delay are calculated. If any bits are set, the packet is discarded; otherwise, the packet header variables are saved.

TEST4 and TEST5

Use for access control and cryptographic authentication. If any bits are set, the packet is discarded immediately and nothing is changed.

TEST6 through TEST8

Check the health of the server. If any bits are set, the packet is discarded; otherwise, the offset and delay relative to the server are calculated and saved.

TEST9

Check the health of the association itself. If any bits are set, the packet is discarded. Otherwise, the saved variables are passed to the clock filter and mitigation algorithms.

TEST10 through TEST12

Check the authentication state using Autokey public-key cryptography. If any bits are set and the association has previously been marked reachable, the packet is discarded; otherwise, the originate and receive timestamps are saved, as required by the NTP protocol, and processing continues.

The *flash* bits for each test are defined as follows:

0x001 TEST1

Copyright © 2014, QNX Software Systems Limited

Duplicate packet. The packet is at best a casual retransmission and at worst a malicious reply.

0x002 TEST2

Bogus packet. The packet is not a reply to a message previously sent. This can happen when the NTP daemon is restarted before somebody else notices.

0x004 TEST3

Unsynchronized. One or more timestamp fields are invalid. This normally happens when the first packet from a peer is received.

0x008 TEST4

Access is denied.

0x010 TEST5

Failure of cryptographic authentication.

0x020TEST6

Server is unsynchronized. Wind up its clock first.

0x040 TEST7

Server stratum is at the maximum of 15. It is probably unsynchronized and its clock needs to be wound up.

0x080 TEST8

Root delay or dispersion is greater than one second, which is highly unlikely unless the peer is unsynchronized.

0x100 TEST9

Peer delay or dispersion is greater than one second, which is highly unlikely.

0x200 TEST10

Autokey protocol has detected an authentication failure.

0x400 TEST11

Autokey protocol has not verified the server or peer.

0x800 TEST12

A protocol or configuration error has occurred in the public key algorithms or a possible intrusion event has been detected.

Copyright © 2014, QNX Software Systems Limited

Caveats:

The `peers` command is nonatomic and may occasionally result in spurious error messages about invalid associations. Also, you wait a long time for timeouts, because the timeout time is a fixed constant, and it assumes the worst-case scenario. In addition, the program doesn't estimate timeout as it sends queries to a particular host.

Copyright © 2014, QNX Software Systems Limited

ntptrace

Trace a chain of NTP servers

Syntax:

```
ntptrace [-dnv] [-r retries] [-t timeout] [server]
```

Runs on:

QNX Neutrino

Options:

-d

Turn on debugging output.

-n

Print only the host IP addresses, not the host names.

-r *retries*

Set the number of retransmission attempts for each host. The default value is 5.

-t *timeout*

Set the value of retransmission timeout in seconds. The default value is 2.

-v

Print verbose information about the NTP servers.

Description:

The `ntptrace` utility determines the source of time for the NTP (Network Time Protocol) servers. It follows the chain of NTP servers back to their master time source. If you don't specify a server, it starts with the `localhost`.

```
$ ntptrace
```

Here's an example of the diagonistic output from the `ntptrace` utility:

```
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2ozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid 'WWVB'
```

On each line, the fields are printed from left to right: the host name, the host stratum, the time offset between that host and the local host, the host synchronization distance, and the reference clock ID. Note that all times are expressed in seconds, and the time

Copyright © 2014, QNX Software Systems Limited

offset mentioned is not always zero. The *stratum* is the server hop count to the primary source, while the *synchronization distance* is the estimated error relative to the primary source. These terms are precisely defined in *RFC 1305*.

Based on:

RFC 1305

Caveats:

The `ntptrace` utility can't improve accuracy by doing multiple samples.

Copyright © 2014, QNX Software Systems Limited

Chapter 16
O

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “O”.

Copyright © 2014, QNX Software Systems Limited

objcopy

Copy the contents of one object file to another (GNU)

Syntax:

objcopy_variant [*options*] *infile* [*outfile*]

where *objcopy_variant* depends on the target platform, as follows:

Target platform	*objcopy_variant*
ARMv7	`ntoarmv7-objcopy`
x86	`ntox86-objcopy`

Runs on:

Linux, Microsoft Windows

Description:

The GNU `objcopy` utility copies the contents of one object file, *infile*, to another, *outfile*. The `objcopy` utility uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file. The exact behavior of `objcopy` is controlled by command-line options.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

objdump

Display information about one or more object files (GNU)

Syntax:

objdump_variant [*options*] *objfile*…

where *objdump_variant* depends on the target platform, as follows:

Target platform	*objdump_variant*
ARMv7	`ntoarmv7-objdump`
x86	`ntox86-objdump`

Runs on:

Linux, Microsoft Windows

Description:

The `objdump` utility displays information about one or more object files. The options control what particular information to display. The *objfile*... arguments identify the object files to be examined. When you specify archives, `objdump` shows information on each of the member object files.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

od

Dump a file in various formats (POSIX)

Syntax:

```
od [-v] [-A format] [-t fmtstr]  [-N count]
   [-j skip] [file]...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-A *format*

Display the file offset field in one of the following formats:

- d — decimal, 9 digits
- n — none (omit this field)
- o — octal, 10 digits
- x — hexadecimal, 7 digits.

The default is o.

-j *skip*

Ignore the first *skip* bytes of data. You can add a trailing character to specify units of blocks (b), kilobytes (k), or megabytes (m).

-N *count*

Display only *count* bytes of input. You can add a trailing character to specify units of blocks (b), kilobytes (k), or megabytes (m).

-t *format*

Display data field using this format specification; see "*Output formats* (p. 1409)," below.

The default format is oS.

-v

Be verbose. If you don't specify the -v option, od folds multiple identical lines into a single line that contains an asterisk (*).

Copyright © 2014, QNX Software Systems Limited

file

The pathname of an input file. If you don't specify any files, od reads from standard input.

Description:

Use the od utility to display a file in various forms, including decimal, hex, octal, and ASCII. The name "od" (octal dump) is derived from the default output format.

The od utility processes input in 16-byte units that are formatted into a line. In the default output format:

- the file offset field is displayed in octal, 10 digits
- a space separates the file offset field from the data
- the data is displayed as four space-separated objects in octal

For example:

```
$ echo "abcdefghijklmnopqrstuvwxyz01234" | od
0000000000 14430661141 15031663145 15432665151 16033667155
0000000020 16434671161 17035673165 06114075171 01215031462
0000000040
```

To exclude part of the input, use the -N and -j options. You can specify the arguments to these options in hex (using a `0x` prefix) or octal (using a `0` prefix). The default units for these options are bytes, but you can specify different units as follows:

To specify:	Add this suffix:
Blocks (512 bytes)	`b`
Kilobytes (1024 bytes)	`k`
Megabytes (1048576 bytes)	`m`

Output formats

To specify the output format, use the -t option. The *format* argument — which you can specify in decimal, hex, or octal — tells od which format to use for presenting the output:

a

Named characters. Display printable characters as themselves, and nonprintable characters as a single dot (`.`).

c

Characters. Display printable characters as themselves; display all other characters as 2-digit hex values, except for the following:

Copyright © 2014, QNX Software Systems Limited

ASCII mnemonic	Value	Representation
NUL	00	\0
<alert>	07	\a
<backspace>	08	\b
<tab>	09	\t
<newline>	0a	\n
<vertical tab>	0b	\v
<formfeed>	0c	\f
<carriage return>	0d	\r

d[1|2|4|C|S|I|L]

Decimal, in objects the size of an `int` by default.

f[4|8|F|D|L]

Floating point, in objects the size of an `float` by default.

o[1|2|4|C|S|I|L]

Octal, in objects the size of an `int` by default.

u[1|2|4|C|S|I|L]

Unsigned decimal, in objects the size of an `int` by default.

x[1|2|4|C|S|I|L]

Hexadecimal, in objects the size of an `int` by default.

The input, processed in 16-byte units formatted into a line, is displayed according to the size you choose:

To display input as:	Choose:
Sixteen 1-byte objects	1
Eight 2-byte objects	2
Four 4-byte values per line	4
Two 8-byte values per line	8
`char`	C
`double`	D

Copyright © 2014, QNX Software Systems Limited

To display input as:	Choose:
`float`	F
`int`	I
`long` or `long double` (depending on the format)	L
`short`	S

Examples:

Display the second to eleventh sectors of the hard disk, `/dev/hd0`:

```
od -j 1b -N 10b /dev/hd0
```

Exit status:

0

All input files were processed successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

omshell

Connect, query, and change ISC DHCP server's state

Syntax:

```
omshell
```

Runs on:

QNX Neutrino

Options:

See the subcommands below.

Description:

The OMAPI Command Shell, `omshell`, provides an an interactive way to connect, query, and possibly change, the ISC DHCP Server's state via OMAPI, the Object Management API. By using OMAPI and `omshell`, you don't have to stop, make changes, and then restart the DHCP server; you can make the changes while the server is running. The `omshell` utility provides a way of accessing OMAPI.

OMAPI is simply a communications mechanism that lets you manipulate objects. To actually use `omshell`, you must understand what objects are available and how to use them.

Documentation for OMAPI objects can be found in the description for servers that provide them (e.g. in the documentation for *`dhcpd`* (p. 552) and *`dhclient`* (p. 478)).

Local and remote objects

Throughout this document, there are references to local and remote objects. Local objects are ones created in `omshell` with the `new` command. Remote objects are ones on the server: leases, hosts, and groups that the DHCP server knows about. Local and remote objects are associated together to enable viewing and modification of object attributes. Also, new remote objects can be created to match local objects.

Opening a connection

You start `omshell` from the command line. Once `omshell` is started, there are several commands that can be issued:

`server` ***address***

Connect to the DHCP server with the given IP address. If this is not specified, the default server is 127.0.0.1 (localhost).

Copyright © 2014, QNX Software Systems Limited

port *number*

Specify the port for OMAPI to listen on. By default, this is 7911.

key *name secret*

Specify the TSIG key to use to authenticate the OMAPI transactions. The *name* is the name of a key defined in *dhcpd.conf* (p. 566) with the `omapi-key` statement. The *secret* is the secret key generated from `dnssec-keygen` or another key-generation program.

connect

Start the OMAPI connection to the server as specified by the `server` statement.

Creating local objects

Any object defined in OMAPI can be created, queried, and/or modified. The object types available to OMAPI are defined in *dhcpd* (p. 552) and *dhclient* (p. 478). When you're using `omshell`, objects are first defined locally, manipulated as desired, and then associated with an object on the server. Only one object can be manipulated at a time. To create a local object, use:

```
new object-type
```

where *object-type* is one of `group`, `host`, or `lease`.

At this point, you have an object that you can set properties on. For example, if you created a new lease object was `new lease`, you can set any of a lease's attributes as follows:

```
set attribute-name = value
```

Attribute names are defined in *dhcpd* (p. 552) and *dhclient* (p. 478). Values should be quoted if they're strings. So, to set a lease's IP address, you would do the following:

```
set ip-address = 192.168.4.50
```

Associating local and remote objects

At this point, you can query the server for information about this lease, by typing:

```
open
```

Now, the local lease object you created and set the IP address for is associated with the corresponding lease object on the DHCP server. All of the lease attributes from the DHCP server are now also the attributes on the local object, and will be shown in `omshell`.

Viewing a remote object

To query a lease of address 192.168.4.50, and find out its attributes, after connecting to the server, take the following steps:

Copyright © 2014, QNX Software Systems Limited

1. `new lease`

   This creates a new local lease object.

2. `set ip-address = 192.168.4.50`

   This sets the local object's IP address to be 192.168.4.50.

3. `open`

   Now, if a lease with that IP address exists, you will see all the information the DHCP server has about that particular lease. Any data that isn't readily printable text will show up in colon-separated hexadecimal values.

In this example, output back from the server for the entire transaction might look like this:

```
> new "lease"
obj: lease
> set ip-address = 192.168.4.50
obj: lease
ip-address = c0:a8:04:32
> open
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "wendelina"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
tsfp = 00:00:00:00
cltt = 00:00:00:00
```

As you can see here, the IP address is represented in hexadecimal, as are the starting and ending times of the lease.

Modifying a remote object

Attributes of remote objects are updated by using the `set` command as before, and then issuing an `update` command. The `set` command sets the attributes on the current local object, and the `update` command pushes those changes out to the server.

Continuing with the previous example, if a `set client-hostname = "something-else"` was issued, followed by an `update` command, the output would look about like this:

```
> set client-hostname = "something-else"
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "something-else"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
```


Copyright © 2014, QNX Software Systems Limited

```
tsfp = 00:00:00:00
cltt = 00:00:00:00
> update
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "something-else"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
tsfp = 00:00:00:00
cltt = 00:00:00:00
```

New remote objects

New remote objects are created much in the same way that existing server objects are modified. Create a local object using `new`, set the attributes as you'd wish them to be, and then create the remote object with the same properties by using:

```
create
```

Now a new object exists on the DHCP server that matches the properties that you gave your local object. Objects created via OMAPI are saved in the *`dhcpd.leases`* (p. 610) file.

For example, to create a new host with the IP address of 192.168.4.40, do the following:

```
> new host
obj: host
> set name = "some-host"
obj: host
name = "some-host"
> set hardware-address = 00:80:c7:84:b1:94
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
> set hardware-type = 1
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 1
> set ip-address = 192.168.4.40
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 1
ip-address = c0:a8:04:28
> create
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
>
```

Your `dhcpd.leases` file would then have an entry like this in it:

```
host some-host {
  dynamic;
  hardware ethernet 00:80:c7:84:b1:94;
  fixed-address 192.168.4.40;
}
```

Copyright © 2014, QNX Software Systems Limited

The dynamic; line is to denote that this host entry did not come from *dhcpd.conf* (p. 566), but was created dynamically via OMAPI.

Resetting attributes

If you want to remove an attribute from an object, you can do this with the unset command. Once you've unset an attribute, you must use the update command to update the remote object. So, if the host somehost from the previous example will not have a static IP address anymore, the commands in omshell would look like this:

```
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
> unset ip-address
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = <null>
>
```

Refreshing objects

A local object may be refreshed with the current remote object properties using the refresh command. This is useful for objects that change periodically, such as leases, to see if they have been updated. This isn't particularly useful for hosts.

Deleting objects

Any remote object that can be created can also be destroyed. This is done by creating a new local object, setting attributes, associating the local and remote object using open, and then using the remove command. If the host some-host from before was created in error, this could be corrected as follows:

```
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
> remove
obj: <null>
>
```

Help

The help command displays all of the commands available in omshell, with some syntax pointers.

Contributing author:

omshell was written by Ted Lemon of Nominum, Inc. Information about Nominum can be found at *http://www.nominum.com* This preliminary documentation was written by Wendy Verschoor of Nominum, Inc., while she was testing omshell.

Copyright © 2014, QNX Software Systems Limited

on

Execute a command on another node or tty (QNX Neutrino)

Syntax:

```
on [-A ability-spec] [-ad | -ae] [-C cpunum] [-d] [-E] [-e
key=value]
   [-h] [-L rlimit:cur[:max]] [-n|f nodename] [-P]
   [-p priority[policy]] [-R runmask] [-s]  [-t tty]
   [-u uid[:gid[,gid,…] | -u user_name | -l user_name]
   [-W nsec[:msec]] [-w device] [-Xsched_command] [command
[args]]
```

Runs on:

QNX Neutrino

Options:

-A ability-spec

Specify an ability to be allowed or disallowed. The *ability-spec* argument is a comma-separated list that contains the following, as required, ignoring the case of the strings:

- the ability identifier, as defined in <sys/procmgr.h>, but omitting the PROCMGR_AID_ prefix (e.g., specify setuid for PROCMGR_AID_SETUID)
- allow or deny
- lock if you want to prevent the process from changing the ability
- root, nonroot, or all to specify the applicable domain
- inherit or noinherit

If the ability accepts a subrange, the above may be followed by a colon and a comma-separated list of subranges, in one of the following forms:

- two numbers separated by a hyphen (e.g., 4-27)
- one number followed by a hyphen (e.g., 4- indicates 4 and greater)
- a single number

You can specify multiple -A options. For example, to deny forking while running as root, but allow the process to set _CS_HOSTNAME when non-root, specify:

```
on -A root,deny,fork -A nonroot,allow,confset:2 ...
```

Copyright © 2014, QNX Software Systems Limited

If you specify `allow`, `deny`, `lock`, `root`, `nonroot`, or `all` without an ability name, the action applies to all abilities not specifically mentioned in another -A option.

For more information about abilities, see the entry for *procmgr_ability()* in the QNX Neutrino *C Library Reference*.

-ad | -ae

Disable or enable Address Space Layout Randomization (ASLR). If you don't specify either of these options, ASLR is left as it currently is.

-C *cpunum*

(QNX Neutrino Core OS 6.3.2 or later) Set the CPU affinity to *cpunum*, where the first CPU is 0. You can use this option multiple times. For more information, see "*Setting the runmask* (p. 1422)", below.

-d

Detach *command* from its parent (i.e., sever the parent/child relationship). This is useful for remotely created processes that never exit and that the shell therefore doesn't need to wait for. Unless this option is specified, a network connection is created connecting the parent to the child.

-E

Clear all environment variables, including any preceding -e options.

-e *key=value*

Define the environment variable *key* with value *value* for the child process. You can specify multiple -e options.

-f *nodename*

Spawn from the remote node using the remote node's / as the network root (i.e., search for the executable on the remote node). In contrast, the -n option searches for the executable on the local node. For more details, see "*The -f vs -n option* (p. 1421)", below.

-h

Start *command* in a HELD state. This option is useful for starting up programs with the intention of debugging them. You can also start up several commands in the HELD state, then send them all a signal to start — they'll all start at almost the exact same time, since their load times will have been eliminated.

-L *rlimit*:*cur*[:*max*]

Copyright © 2014, QNX Software Systems Limited

Specify a limit on a system resource for the child process. The arguments are as follows:

- *rlimit* — either a textual string (e.g., `"freemem"`) or a numeric value that identifies the rlimit being set; see `RLIMIT_*` in `<sys/resource.h>`. The case in the textual string is ignored.
- *cur* — the *rlim_cur* field (i.e., the soft limit). It can be a number or `"inf"` for `RLIM_INFINITY`.
- *max* (optional) — the *rlim_max* field (i.e., the hard limit). It can also be a number or `"inf"`. If you don't specify this limit, it maintains its previous value.

For more information about the limits, see the entry for *setrlimit()* in the QNX Neutrino *C Library Reference*.

-l *user_name*

("el") Login as the given user. This option is similar to the -u option, but also sets the ***LOGNAME***, ***HOME***, and ***SHELL*** environment variables, sets the *umask* (p. 2005) to `022`, and changes to the directory specified for the user in the password database.

-n *nodename*

Execute *command* (as found on the local node) on the remote *nodename*. In contrast, the -f option searches for the executable on the remote node. For more details, see "*The -f vs -n option* (p. 1421)," below.

-P

(QNX Neutrino 6.4.0 or later) Spawn the process, setting the SPAWN_PADDR64_SAFE flag to indicate that the process is known to operate safely with 64-bit addressing or doesn't care about the physical memory location.

-p *priority*[*policy*]

Execute the command at the specified priority, optionally changing the scheduling policy.

Priorities are in the range from 0 through 255. Priority 0 is used for the idle thread; by default, priorities of 64 and greater are privileged, so only processes with an effective user ID of 0 (i.e., `root`) can use them. Non-`root` (and `root`) processes can use priorities from 1 through 63.

You can change the range of privileged priorities with the -P option for *procnto* (p. 1586).

The scheduling policy must be one of:

Copyright © 2014, QNX Software Systems Limited

- `f` — FIFO
- `r` — round-robin
- `o` — other (currently the same as round-robin).

If you don't specify a *command*, the change applies to the parent process.

-R *runmask*

(QNX Neutrino Core OS 6.3.2 or later) Set the CPU affinity to *runmask*. You can use this option multiple times to specify masks that are more than 32 bits long. For more information, see "*Setting the runmask* (p. 1422)," below.

-s

Spawn the command in a new process group.

-t *tty*

Open the specified terminal name as file descriptors 0, 1, and 2 for *command*. The *command* is run in a new session with *tty* as its controlling terminal. If *tty* doesn't contain a slash (`/`), `/dev/` is added to the beginning.

-u *uid*[:*gid*[,*gid*,...]] or -u *user_name*

Run as the user specified by the numeric *uid*, in the specified group(s), or as the given *user_name*.

-W *nsec*[:*msec*]

The number of seconds to wait for the device specified in the following -w option. The default is forever.

(QNX Neutrino 6.6 or later) You can optionally specify a polling interval, in milliseconds. The default is 100 milliseconds.

-w *device*

Wait for a *stat()* on the given *device* to succeed before continuing. If *device* doesn't contain a slash (`/`), `/dev/` is added to the beginning.

You can repeat the -w and -W options on the command line. They're processed in the order given, *before any other options*.

-X*sched_cmd*

(QNX Neutrino Core OS 6.3.2 or later) Launch using the specified command for an external scheduler. The possible commands include:

Copyright © 2014, QNX Software Systems Limited

- aps=*partition_name* — launch the application into the adaptive partition with the given name. For more information, see the Adaptive Partitioning *User's Guide*.

***command* [*args*]**

The command to be executed, and any arguments to be passed to it.

Description:

The on utility extends the process creation abilities of the shell (*sh* (p. 1760)). You can start a process on a remote node, on a different controlling terminal, in a HELD state for debugging or later synchronized starting.

If the -d option *isn't* specified, a network connection is created as a local agent for the remote child process. When the child terminates, the parent must do a *wait()* on the created connection to reap the zombie process entry for the child. If the -d option *is* specified, the command is detached from its parent. The parent isn't able to do a *wait()* for the child, nor is it able to control it via signals.

By default, the command is run in the current session. The -t option starts a new session, which means the command won't receive a `SIGHUP` if the current session leader terminates.

The `on -t` command becomes the new session leader on the tty specified; that is, it receives `SIGHUP` generated by hangups on that tty. Any processes originally running on that tty don't get `SIGHUP`, and this condition persists even when the process started by on has terminated. For this reason, specify only ttys that aren't currently in use.

The -f vs -n option

The -f and -n options look similar, but they're subtly different. To illustrate this, let's consider two nodes, `this_node` and `that_node`. On each node, we'll create a copy of the `sleep` command, appending the node name; on the `this_node` node, this looks like this:

```
cp /usr/bin/sleep /tmp/sleep_this_node
```

Now, let's run the `sleep_that_node` command with the -f and -n options:

```
$ on -f /net/that_node /tmp/sleep_that_node 1
$ on -n /net/that_node /tmp/sleep_that_node 1
on: No such file or directory (/tmp/sleep_that_node)
```

The first command succeeds because we're using `/net/that_node` as the network root, and that's where `sleep_that_node` exists. The second command fails because `sleep_that_node` doesn't exist on our local node.

```
$ on -f /net/that_node /tmp/sleep_this_node 1
```

Copyright © 2014, QNX Software Systems Limited

```
on: No such file or directory (/tmp/sleep_this_node)
$ on -n /net/that_node /tmp/sleep_this_node 1
```

This time, the first command fails because `sleep_this_node` doesn't exist on `/net/that_node`. The second command finds `sleep_this_node` on the local node, and executes it on `/net/that_node`.

As a more useful example, let's use *pidin* (p. 1521) to get information about the processes running on `that_node`:

```
on -f /net/that_node pidin | less
```

If you use the -n option instead, like this:

```
on -n /net/that_node pidin | less
```

you get information about the processes running on `this_node`, but the command is executed on `that_node`.

Setting the runmask

On a multicore system, you can use a runmask to specify which processors a thread or process can run on. The default is all 1s (i.e., all CPUs).

The runmask is useful only on multiprocessor systems.

You can use `on` to set the runmask and inherit mask for a new process; to change the masks for threads that are already running, use *slay* (p. 1774). Both commands interpret the -C and -R options in the same way.

You can use more than one -R option to specify a runmask that's more than 32 bits long. The first -R option specifies bits 0 through 31, the second specifies bits 32 through 63, and so on.

If you use both the -C and -R options or multiple instances of them, the resultant mask is the bitwise ORing of all -C and -R options. For example, `on -R 0x1` is equivalent to `on -C0`, and `on -R 0x1 -C3` is equivalent to `on -C0 -C3`. The `on` command sets the process's runmask and inherit mask to the same value.

For more information about runmasks, see the Multicore Processing chapter of the *System Architecture* guide, and the Developing Multicore Systems chapter of the Multicore Processing *User's Guide*.

Examples:

Run *login* (p. 1121) on console 2:

```
on -t con2 login
```

Run *pidin* (p. 1521) on the node named `ruth`:

```
on -n ruth pidin
```


Copyright © 2014, QNX Software Systems Limited

Run *sort* (p. 1826) as an orphan on the node named peter:

```
on -d -n peter sort file.dat
```

Run *pidin* (p. 1521) on node george with a new session, its standard I/O connected to console 1 on node ruth:

```
on -t /net/ruth/dev/con1 -n george pidin
```

Exit status:

The on utility exits with the exit status of *command*.

Copyright © 2014, QNX Software Systems Limited

op

Run a command as someone else

Syntax:

`op` *command*

Runs on:

QNX Neutrino

Options:

None.

Description:

The `op` utility was designed as an easy way to run commands as `root`.

This utility is a security hole. You have to log in as `root` and do the following in order to make it do anything:

```
chmod +s /usr/bin/op
```

You should use *su* (p. 1872) instead.

Copyright © 2014, QNX Software Systems Limited

openssl

Command-line tool for using the OpenSSL crypto library

Syntax:

```
openssl command [command_opts] [command_args]

openssl [list-standard-commands |
        list-message-digest-commands |
        list-cipher-commands |
        list-cipher-algorithms |
        list-message-digest-algorithms |
        list-public-key-algorithms]

openssl no-cmd [arbitrary_options]
```

Runs on:

QNX Neutrino

Options:

None.

Description:

OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards that they require.

The `openssl` program is a command-line tool for using the various cryptography functions of OpenSSL's crypto library from the shell. You can use it for the following:

- creation and management of private keys, public keys and parameters
- public key cryptographic operations
- creation of X.509 certificates, CSRs and CRLs
- calculation of Message Digests
- encryption and Decryption with Ciphers
- SSL/TLS Client and Server Tests
- handling of S/MIME signed or encrypted mail
- timestamp requests, generation and verification

In order for `openssl` to be fully functional, you must have started *random* (p. 1654) with the -t option.

Command summary

Copyright © 2014, QNX Software Systems Limited

The `openssl` program provides a rich variety of commands (*command* in the synopsis above), each of which often has a wealth of options and arguments (*command_opts* and *command_args*).

The pseudo-commands `list-standard-commands`, `list-message-digest-commands`, and `list-cipher-commands` output a list (one entry per line) of the names of all standard commands, message digest commands, or cipher commands, respectively, that are available in the present `openssl` utility.

The pseudo-commands `list-cipher-algorithms` and `list-message-digest-algorithms` list all cipher and message digest names, one entry per line. Aliases are listed as:

from => *to*

The pseudo-command `list-public-key-algorithms` lists all supported public key algorithms.

The pseudo-command `no-`*cmd* tests whether a command of the specified name is available. If no command named *cmd* exists, `openssl` returns 0 (success) and prints `no-`*cmd*; otherwise it returns 1 and prints *cmd*. In both cases, the output goes to *stdout*, and nothing is printed to *stderr*. Additional command-line arguments are always ignored. Since for each cipher there's a command of the same name, this provides an easy way for shell scripts to test for the availability of ciphers in the `openssl` program. (The `no-`*cmd* can't detect pseudo-commands such as `quit`, `list-...-commands`, or `no-`*cmd* itself.)

Standard commands

`asn1parse`

Parse an ASN.1 sequence.

`ca`

Certificate Authority (CA) Management.

`ciphers`

Cipher Suite Description Determination.

`cms`

Cryptographic Message Syntax utility.

`crl`

Certificate Revocation List (CRL) Management.

`crl2pkcs7`

CRL to PKCS#7 Conversion.

Copyright © 2014, QNX Software Systems Limited

dgst

Message Digest Calculation.

dh

Diffie-Hellman Parameter Management; rendered obsolete by `dhparam`.

dhparam

Generation and Management of Diffie-Hellman Parameters. Superseded by `genpkey` and `pkeyparam`.

dsa

DSA Data Management.

dsaparam

DSA Parameter Generation and Management. Superseded by `genpkey` and `pkeyparam`.

ec

Elliptical Curve key processing.

ecparam

EC parameter manipulation and generation.

enc

Encoding with Ciphers.

engine

Engine (loadable module) information and manipulation.

errstr

Error Number to Error String Conversion.

gendh

Generation of Diffie-Hellman Parameters; rendered obsolete by `dhparam`.

gendsa

Generation of DSA Private Key from Parameters. Superseded by `genpkey` and `pkey`.

genpkey

Generation of Private Key or Parameters.

genrsa

Copyright © 2014, QNX Software Systems Limited

Generation of RSA Private Key. Superseded by `genpkey`.

`nseq`

Create or examine a Netscape certificate sequence.

`ocsp`

Online Certificate Status Protocol utility.

`passwd`

Generation of hashed passwords.

`pkcs12`

PKCS#12 Data Management.

`pkcs7`

PKCS#7 Data Management.

`pkey`

Public and private key management.

`pkeyparam`

Public key algorithm parameter management.

`pkeyutl`

Public key algorithm cryptographic operation utility.

`rand`

Generate pseudo-random bytes.

`req`

PKCS#10 X.509 Certificate Signing Request (CSR) Management.

`rsa`

RSA key management.

`rsautl`

RSA utility for signing, verification, encryption, and decryption. Superseded by `pkeyutl`.

`s_client`

This implements a generic SSL/TLS client that can establish a transparent connection to a remote server speaking SSL/TLS. It's intended for testing

Copyright © 2014, QNX Software Systems Limited

purposes only and provides only rudimentary interface functionality but internally uses mostly all functionality of the OpenSSL `ssl` library.

`s_server`

This implements a generic SSL/TLS server that accepts connections from remote clients speaking SSL/TLS. It's intended for testing purposes only and provides only rudimentary interface functionality but internally uses mostly all functionality of the OpenSSL ssl library. It provides both its own command-line-oriented protocol for testing SSL functions and a simple HTTP response facility to emulate an SSL/TLS-aware webserver.

`s_time`

SSL Connection Timer.

`sess_id`

SSL Session Data Management.

`smime`

S/MIME mail processing.

`speed`

Algorithm Speed Measurement.

`spkac`

SPKAC printing and generating utility.

`ts`

Time Stamping Authority tool (client/server).

`verify`

X.509 Certificate Verification.

`version`

OpenSSL Version Information.

`x509`

X.509 Certificate Data Management.

Message digest commands

`md2`

MD2 Digest.

`md5`

Copyright © 2014, QNX Software Systems Limited

MD5 Digest.

mdc2

MDC2 Digest.

rmd160

RMD-160 Digest.

sha

SHA Digest.

sha1

SHA-1 Digest.

sha224

SHA-224 Digest.

sha256

SHA-256 Digest.

sha384

SHA-384 Digest.

sha512

SHA-512 Digest.

Encoding and cipher commands

base64

Base64 Encoding.

bf, **bf-cbc**, **bf-cfb**, **bf-ecb**, **bf-ofb**

Blowfish Cipher.

cast, **cast-cbc**

CAST Cipher.

cast5-cbc, **cast5-cfb**, **cast5-ecb**, **cast5-ofb**

CAST5 Cipher.

des, **des-cbc**, **des-cfb**, **des-ecb**, **des-ede**, **des-ede-cbc**, **des-ede-cfb**, **des-ede-ofb**, **des-ofb**

DES Cipher.

Copyright © 2014, QNX Software Systems Limited

des3, desx, des-ede3, des-ede3-cbc, des-ede3-cfb, des-ede3-ofb

Triple-DES Cipher.

idea, idea-cbc, idea-cfb, idea-ecb, idea-ofb

IDEA Cipher.

rc2, rc2-cbc, rc2-cfb, rc2-ecb, rc2-ofb

RC2 Cipher.

rc4

RC4 Cipher.

rc5, rc5-cbc, rc5-cfb, rc5-ecb, rc5-ofb

RC5 Cipher.

Pass phrase arguments

Several commands accept password arguments, typically using `-passin` and `-passout` for input and output passwords respectively. These allow the password to be obtained from a variety of sources. Both of these options take a single argument whose format is described below. If no password argument is given and a password is required, you're prompted to enter one: this will typically be read from the current terminal with echoing turned off.

pass:*password*

The actual password is *password*. Since the password is visible to utilities, you should use this form only where security isn't important.

env:*var*

Obtain the password from the environment variable *var*. Since the environment of other processes is visible on certain platforms, you should use this option with caution.

file:*pathname*

The first line of *pathname* is the password. If you supply the same *pathname* argument to `-passin` and `-passout` arguments, the first line is used for the input password, and the next line for the output password. The *pathname* need not refer to a regular file; it could, for example, refer to a device or named pipe.

fd:*number*

Read the password from the given file descriptor number. You can use this, for example, to send the data via a pipe.

Copyright © 2014, QNX Software Systems Limited

stdin

Read the password from standard input.

Exit status:

0

Success.

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 17
P

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`portmap`	*`rpcbind`* (p. 1695)
`psin`	*`pidin`* (p. 1521)

This chapter describes the utilities, etc. whose names start with “P”.

Copyright © 2014, QNX Software Systems Limited

passwd

Change the login password or create new user names (UNIX)

Syntax:

```
passwd [-dmSs] [-t iterations] [-w width] [name]
```

Runs on:

QNX Neutrino

Options:

-d

Use the DES password hash.

-m

Use the MD5 password hash.

-S

Use the SHA-512 password hash (the default).

-s

Use the SHA-256 password hash.

-t *iterations*

The number of times to iterate the hash. The default is 1000.

-w *width*

The width of the salt, in multiples of 8 bytes. The minimum is 8, and the default is 16.

name

The username whose password is to be changed or for whom an account is to be created (`root` only).

Description:

You can use the `passwd` utility to change your login password, and if you're logged in as the superuser (`root`), you can create a new user account.

Copyright © 2014, QNX Software Systems Limited

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

If you're changing your password, `passwd` prompts for the old password and then for the new password. The new password must be entered twice, to avoid typing mistakes. Only the owner or the superuser may change a password.

To create a new user account, type:

```
passwd new_user_name
```

Make sure that the user name is no longer than 14 characters; otherwise, that user won't be able to log in.

passwd file

When creating a new user account, `passwd` prompts for information, such as the user's group list, home directory, and shell. The `/etc/default/passwd` file (see "*Files* (p. 1435)," below) specifies the default values for these prompts. You can edit this file to modify `passwd`'s behavior to suit local requirements.

The `/etc/passwd` file contains the following fields, separated by colons:

```
username:has_passwd:userid:groupid:misc:home_directory:initial_command
```

If the *has_passwd* field contains an `x` character, a password has been defined for this user. If no character is present, no password has been defined. Use of any other character is reserved and may cause side-effects for the user.

The *groupid* field contains a group number. Users may log in under the *groupid* listed in their `/etc/passwd` file entry without being listed as a member of that group in the `/etc/group` file.

The *misc* field stores supplemental information, with commas separating subfields. Usually, the first subfield contains the user's "real life" name. Some utilities use this information.

The *initial_command* field contains the initial command to run after the user has successfully logged in. This command and any arguments it takes must be separated by tab or space characters. As the command is spawned directly (not run by a shell), no shell expansions is performed. There is no mechanism for specifying command-line arguments that contain space or tab characters themselves. (Quoting isn't supported.)

If no *initial_command* is specified, */bin/sh* (p. 1760) is used.

Files:

Copyright © 2014, QNX Software Systems Limited

/etc/.pwlock

This file is created by passwd to indicate to other instances of passwd that the password file is currently being modified. When passwd finishes, the file is removed. See "Caveats," below.

/etc/group

This file defines the known groups for the system. It associates group *names* with a numerical *ID* and a list of *username*s who are members of the group.

Entries in this file appear in the following format:

```
groupname:x:groupid:user[,user]...
```

The x represents the group password; QNX Neutrino doesn't support group passwords.

/etc/passwd

Contains the user account entries. The format of entries in this file are as follows:

```
username:has_passwd:userid:groupid:misc:home_directory:initial_command
```

/etc/shadow

Contains encoded versions of the actual passwords for user accounts. The passwords themselves aren't stored in the /etc/passwd file.

/etc/opasswd, /etc/oshadow

When passwd modifies a password file, it first locks the password files with the /etc/.pwlock file, then copies the contents of the current /etc/passwd and /etc/shadow files to /etc/opasswd and /etc/oshadow, respectively. If passwd is killed before it finishes writing the updated file, the password files may be restored from these backup versions. See "Caveats," below.

/etc/default/passwd

Specifies the settings that the passwd utility uses when you create a new user account. If you're the system administrator, you can edit this file. The settings include the following, shown with the value specified by default in this file:

Setting	Default	Description
BASEDIR=*dirname*	/home	The base directory under which user directories are created.

Copyright © 2014, QNX Software Systems Limited

Setting	Default	Description
SHELL=*progname*	/bin/sh	The shell to use for the login shell field in new password entries.
UIDRANGE=*low*-[*high*]	100-	The valid range of values for new user IDs. You can omit the *high* component, indicating no upper bound, but you still need the dash.
GIDRANGE=*low*-[*high*]	100-	The valid range of values for group IDs. As with UIDRANGE, you can omit the *high* component if there's no upper bound.
DUPDIROK	Not set.	If specified, passwd lets you select an existing directory as a new user's home directory.
DUPUIDOK	Not set.	If specified, passwd lets you select an existing user ID for a new user name. This is generally discouraged, because it allows many user names to be mapped to one user ID.
NOPASSWORDOK	NOPASSWORDOK	If specified, passwd lets you set up user accounts that don't require a password to log in.
STRICTPASSWORD	Not set.	If specified, passwd requires all passwords to contain at least two types of characters (e.g.

Copyright © 2014, QNX Software Systems Limited

Setting	Default	Description
		alphabetic and punctuation).
INSISTANT=*retries*	6	The number of times passwd asks non-root users if they really want to set up their account with no password. This variable is ignored if NOPASSWORDOK is set.
PROFILE=*basename*	.profile	The name to use for the shell's initialization file in the user's home directory. The file specified by DEFPROFILE (below) is copied there when you set up a new account.
DEFPROFILE=*filename*	/etc/skel/.profile	The path to a default shell-initialization file that's copied to a new user's PROFILE when you set up the account.
QNXCRYPT	Not set.	If this is set, passwd uses the old QNX 4 encryption method, instead of the default UNIX encryption method.

Caveats:

The passwd utility creates the /etc/.pwlock file during updates to the password database. If for some reason the system crashes at an inopportune moment and leaves this file present, passwd will refuse to work until the file is removed by the system administrator. If the password files are somehow left in an inconsistent state as a result of the crash, the system administrator should also copy /etc/oshadow to /etc/shadow and copy /etc/opasswd to /etc/passwd.

Copyright © 2014, QNX Software Systems Limited

paste

Merges lines of given input files, and writes the resulting lines to standard output. (POSIX)

Syntax:

```
paste [-s] [-d list] file
```

Runs on:

QNX Neutrino

Options:

-d *list*

Specifies that each character in the *list* is an element specifying a delimiter character. Unless a backslash character appears in list, each character in list is an element specifying a delimiter character. If a backslash character appears in list, the backslash character and one or more characters following it are an element specifying a special delimiter character, described below. These elements specify one or more delimiters to use, instead of the default tab, to replace the newline of the input lines. If the -s option is also specified:

- The last <newline> in a file isn't modified.
- The delimiter is reset to the first element of the *list* after each file operand is processed.

If the -s option is not specified:

- The <newline>s in the file specified by the last file operand can't be modified.
- The delimiter is reset to the first element of the *list* each time a line is processed from each file.

When a backslash character appears in *list*, the backslash character and the other characters that follow it are used to represent these delimiter characters:

- \n represents a <newline>
- \t represents a <tab>
- \\ represents a backslash character
- \0 represents an empty string, not a NULL character. If a lower case or upper case “x” immediately follows `\0`, or any character defined by the

Copyright © 2014, QNX Software Systems Limited

LC_CTYPE digit keyword, the results are unspecified. For more information, see the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 7, Locale.

When the escape sequences of the *list* argument are used in a shell script, you must use single quotation marks; otherwise, the shell interprets the backslash character as a special character.

The input files must be text files.

-s

Concatenates all lines of each separate input file in command line order. Except for the last line in each input file, the <newline> of every line is replaced with <tab>, unless otherwise specified by the -d option.

file

The pathname of an input file. Specifying "-" for one or more of the files uses the standard input, which is input read one line at a time for each instance of "-".

Description:

For every file you name, the `paste` utility pastes columns or fields from each line, concatenates them, and writes them to the standard output.

By default, `paste` assumes the field delimiter character to be tab. You can use the -d option to specify another delimiter.

Examples:

The following are examples of the *list* argument:

- Write out a directory in three columns:

```
ls | paste - - -
```

- Combine pairs of lines from a file into single lines:

```
paste -s -d "\t\n" file
```

Exit status:

0

Successful completion.

Copyright © 2014, QNX Software Systems Limited

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

pathtrust

Designate a file or filesystem as trusted, or see if it is

Syntax:

Mark a file or filesystem as trusted:

```
pathtrust [!]file... [lockdown]
```

Test to see if a file is trusted:

```
pathtrust [-q] -t file... [lockdown]
```

Runs on:

QNX Neutrino

Options:

-q

Be quiet; use only the return code to indicate whether or not the file is trusted.

-t

Test to see if the file is trusted. If you haven't also specified -q, `pathtrust` reports the results on standard output.

[!]*file*

The file to test or mark as trusted.

If you haven't specified the -t option, then if you specify a leading exclamation mark, the given file is designated as trusted. If you don't specify the exclamation mark, the underlying filesystem is designated as trusted.

lockdown

Prevent any other files or filesystems from being marked as trusted.

Description:

The `pathtrust` utility sends messages to *procnto* (p. 1586) to mark the given files and filesystems as trusted. If you don't mark any files or filesystems as trusted, all are trusted.

Copyright © 2014, QNX Software Systems Limited

If a process with any privileged abilities enabled attempts to mark a region of memory as PROT_EXEC, any memory-mapped files in the region must be trusted or be from a trusted filesystem. For more information about abilities, see *procmgr_ability()* in the QNX Neutrino *C Library Reference*.

Exit status:

0

Successful completion; the file or filesystem is trusted.

1

The file or filesystem isn't trusted.

2

An error occurred.

Copyright © 2014, QNX Software Systems Limited

pax

Portable archive interchange (POSIX)

Syntax:

List archive contents:

```
pax [-cimopuvy] [-f archive] [-s replstr]...
    [-t device] [pattern...]
```

Read an archive:

```
pax -r [-cimnopuvy] [-f archive] [-s replstr]...
    [-t device] [pattern...]
```

Write an archive:

```
pax -w [-dimuvy] [-b blocking] [-[a]f archive]
    [-s replstr]... [-t device] [-x format]
    [pathname...]
```

Copy files:

```
pax -rw [-ilmopuvy] [-s replstr]... [pathname...]
    directory
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Append the files specified by *pathname* to the archive specified with -f.

-b *blocking*

Block the output at *blocking* bytes per write to the archive file. A **k** suffix multiplies blocking by 1024, a **b** suffix multiplies blocking by 512, and an **m** suffix multiplies blocking by 1048576 (1 megabyte). If not specified, blocking is automatically determined on input and is ignored for -rw (copy files).

-c

Complement the match sense of the *pattern* operands.

-d

Copyright © 2014, QNX Software Systems Limited

Don't create intermediate directories not explicitly listed in the archive. This option is applied only if you specify the -r option.

-f *archive*

Use *archive* as the pathname of the input or output archive, overriding the default of standard input for -r or standard output for -w.

-i

Interactively rename files. Substitutions specified by -s options are performed before requesting the new filename from you. If you enter an empty line, the file is skipped. If EOF is encountered, pax exits with an exit status of 0.

-l

("el") When possible, link rather than copy files.

-m

Don't keep file modification times.

-n

When you specify -r, but not -w, treat the *pattern* operands as ordinary filenames. Only the first occurrence of each of these files in the input archive is read. The pax utility exits with an exit status of 0 after all files in the list have been read. If it can't find one or more of the files in the list, pax writes a diagnostic to standard error for each of these files and exits with a nonzero exit status. The filenames are compared before any of the -i, -s, or -y options are applied.

-o

Restore file ownership as specified in the archive. The invoking process must have appropriate privileges to accomplish this.

-p

Preserve the access time of the input files after they have been copied.

-s *replstr*

Modify filenames according to the substitution expression. The syntax for the expression is:

-s /*old*/*new*/[gp]

You can use any non-null character as a delimiter (a / is used here as an example). Multiple -s expressions are applied in the order specified terminating with the first successful substitution. The optional trailing p

Copyright © 2014, QNX Software Systems Limited

causes successful mappings to be listed on standard error. The optional trailing g causes the old expression to be replaced each time it occurs in the source string. If a filename becomes an empty string after applying substitutions to it, it's ignored both on input and output.

-t *device*

The *device* argument names the input or output archive device, overriding the default of standard input for -r and standard output for -w.

-u

Copy each file only if it is newer than a preexisting file with the same name.

-v

Be verbose; list filenames as they're encountered. This option produces a table of contents listing on the standard output when both -r and -w are omitted; otherwise the filenames are printed to standard error as they are encountered in the archive.

-x *format*

Use this output archive format. The input format is automatically determined when you use the -r option. The supported formats are:

cpio

The extended *cpio* (p. 151) interchange format specified in POSIX Std 1003.1-1988.

ustar

The extended *tar* (p. 1890) interchange format, also specified in POSIX Std 1003.1-1988. This is the default archive format.

-y

Interactively prompt for the disposition of each file. Substitutions specified by -s options (described above) are performed before you're prompted for the disposition. EOF or an input line starting with the character q causes pax to exit. Otherwise, an input line starting with anything other than y causes the file to be ignored. You can't use this option in conjunction with the -i option.

Only the last -f or -t option takes effect.

Copyright © 2014, QNX Software Systems Limited

directory

The destination directory pathname for copies when both the -r and -w options are specified. The directory must exist and you must have the appropriate write permissions, or an error results.

pathname

A file to be copied or a directory containing files and subdirectories to be (recursively) copied in addition to the directory itself.

pattern

A pattern given in the standard shell pattern-matching notation. If no pattern is specified, the default is *, which selects all files.

Modes of operation:

If you don't specify -r or -w, then pax lists the contents of the specified archive. In this mode, pax lists normal files one per line. Hard link pathnames are listed as:

```
pathname == linkname
```

where *pathname* is the name of the file being extracted, and *linkname* is the name of a file that appeared earlier in the archive.

Symbolic link pathnames are listed as:

```
pathname -> destination_path
```

If you specify -v, then pax lists normal pathnames in the same format used by the *ls* (p. 1139) utility with the -l ("el") option, except for hard links, which are shown as:

```
<ls -l listing> == linkname
```

The modes of operation related to combinations of -r and -w are as follows:

-r

Read an archive file from the standard input; select for extraction only those files whose names match any of the *pattern* operands. The selected files are conditionally created and copied relative to the current directory tree, subject to the options chosen. By default, the owner and group of selected files are those of the invoking process, and the permissions and modification times are the same as those in the archive.

The supported archive formats are automatically detected on input.

-w

Copyright © 2014, QNX Software Systems Limited

Write the files and directories specified by *pathname* operands to the standard output, together with the pathname and status information prescribed by the archive format used. The default output format is `tar`, but you can override this by using the -x format option described below.

A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, then the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied.

-rw

Read the files and directories named in the *pathname* operands and copy them to the destination directory. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied. The directory named by the directory operand must exist and must have the proper permissions before the copy can occur.

Description:

The `pax` utility reads and writes archive files that conform to the archive/interchange file format specified in POSIX Std 1003.1-1988. The utility can also read, but not write, a number of other file formats. Support for these traditional file formats (such as V7 `tar` and System V binary `cpio` format archives) is provided for backward compatibility and to maximize portability.

The `pax` utility also supports traditional `cpio` and System V `tar` interfaces if invoked with those names (they're links to `pax`).

The `pax` utility is capable of reading and writing archives that span multiple physical volumes. Upon detecting an end of medium on an archive that isn't yet completed, `pax` prompts you for the next volume of the archive and lets you specify the location of the next volume.

If the `pax` archive is stored directly in a floppy disk block special file (e.g. `/dev/fd0`), the archive overwrites the first block of the disk, which the floppy driver, *devb-fdc* (p. 244), uses to store media-type information that lets it dynamically adjust to diskettes of differing media capacity being inserted in the drive (e.g. 720 KB vs 1.4 MB, 360 KB vs 1.2 MB etc.). If the floppy driver can't determine the capacity, it assumes 1.4 MB.

Copyright © 2014, QNX Software Systems Limited

Combinations of the -r and -w command-line arguments specify whether pax reads, writes, or lists the contents of the specified archive, or moves the specified files to another directory.

When writing to an archive, the standard input is used as a list of pathnames if no *pathname* operands are specified. The format is one pathname per line. When reading, the standard input is the archive file, which is formatted according to one of the format specifications in POSIX Std 1003.1-1988.

The user ID and group ID of the process, together with the appropriate privileges, affect the ability of pax to restore ownership and permissions attributes of the archived files. (See Archive/Interchange File Format in POSIX Std 1003.1-1988.)

Note that the options -a, -c, -d, -i, -l, -p, -t, and -y are provided for functional compatibility with the historical cpio and tar utilities. The option defaults were chosen based on the most common usage of these options, so some of the options have meanings different from those of the historical commands.

Examples:

Copy the contents of the current directory to the floppy drive:

```
pax -w -f /dev/fd0 .
```

Copy the contents of olddir to newdir:

```
mkdir newdir
cd olddir
pax -rw . ../newdir
```

Read the archive pax.out with all files rooted in /usr in the archive extracted relative to the current directory (note the use of commas as pattern separators for the -s option):

```
pax -r -s ",^/usr/,," -f pax.out
```

Files:

The controlling terminal (/dev/tty) is used to prompt the user for information when the -i or -y options are specified.

Exit status:

0

All files in the archive were processed successfully.

>0

The pax utility aborted due to errors encountered during operation.

Copyright © 2014, QNX Software Systems Limited

Caveats:

Special permissions may be required to copy or extract special files.

Device, user ID, and group ID numbers larger than 65535 cause additional header records to be output. These records are ignored by some historical versions of `cpio` and `tar`.

The archive formats described in Archive/Interchange File Format have certain restrictions that have been carried over from historical usage. For example, pathnames stored in the archive can be no more than 255 characters in length.

When getting an `ls -l` style listing on `tar` format archives, link counts are listed as zero since the `ustar` archive format doesn't keep link count information.

On 16-bit architectures, including 16-bit versions of QNX 4., the largest buffer size is 32K-1. This is due, in part, to using integers in the buffer allocation schemes. On many of these machines, however, it isn't possible to allocate blocks of memory larger than 32K.

Copyright © 2014, QNX Software Systems Limited

pccard-launch

Automatically start drivers when PC Cards are inserted

You must be `root` to run this utility.

Syntax:

```
pccard-launch [-nv] 'type,command'...
```

Runs on:

QNX Neutrino

Options:

-n

Don't lock the socket where the expected card is found.

-v

Display verbose messages on standard output.

'type,command'

For this PC Card type, spawn *command*. Types may be specified in decimal (*nnnn*) or hexadecimal (`0x`*nnn*), and are followed by a comma (`,`) and the command to execute. For example:

```
pccard-launch '0x600,io-pkt-v4 -dne2000 ioport=0x$IOPORT,\
irq=$IRQ'
```

You can specify multiple type/command pairs on the command line by separating them with whitespace. If the command specified contains whitespace, the command must be enclosed in quotes (as shown).

Description:

The `pccard-launch` utility automatically starts drivers and sets environment variables describing the ports and IRQs used by a card, thus permitting a non-PC Card-aware driver to be started automatically when a card is inserted, and stopped when removed.

The environment variables may be named in the command lines for the drivers to be run, thus allowing the port and IRQ information to be passed on the command line to

Copyright © 2014, QNX Software Systems Limited

the drivers, most of which aren't aware of the environment variables set by `pccard-launch`.

This utility both starts driver when cards are plugged in and stops them via a `SIGTERM` signal when cards are removed.

When a card is plugged in, `pccard-launch` checks the card type against the list of card types/driver commands supplied to it on its command line. If it doesn't know about the card type, it does nothing. If it has been provided with a command for the card type, it does the following:

- Locks the card.
- Sets up environment variables detailing I/O port locations, IRQ used, etc.
- Spawns the associated command.

When the card is removed, `pccard-launch` unlocks the card and signals the previously spawned process with `SIGTERM`.

You can determine the card type by using the *pin* (p. 1541) command. A `pin con` command displays configuration information for the PC Card. Look at the `config` line, which should look something like:

```
config = 0xNN, 0xTTTT..."
```

where *NN* is the configuration number and *TTTT* is the type of the card. For example, a modem card has a type of `0x200`. This is the type that you must use on the command line for `pccard-launch`.

Examples:

Automatically start the ne2000 driver with the appropriate I/O port and IRQ settings, for PC Card type 0x600:

```
pccard-launch '0x600,io-pkt-v6-hc -dne2000 ioport=0x$IOPORT,irq=$IRQ'
```

Files:

`pccard-launch`

This utility is located in the `/usr/sbin/` directory.

Environment variables:

The environment variables set by `pccard-launch` are:

IOPORT

Hex address of the I/O port (e.g. 320).

IOPORTSZ

Copyright © 2014, QNX Software Systems Limited

Size of the I/O port (e.g. 32).

IOPORT2

Hex address of the second I/O port, if assigned.

IOPORT2SZ

Size of the second I/O port, if assigned.

IRQ

IRQ of the device.

SOCKET

The socket where the card is inserted.

Copyright © 2014, QNX Software Systems Limited

pci

Display PCI devices

You need to log in as `root` to use this utility.

Syntax:

```
pci [-n] [-v]
```

Runs on:

QNX Neutrino

Options:

-n

The maximum number of PCI buses to scan.

-v

Be verbose; display all PCI devices.

Description:

The `pci` utility displays Peripheral Component Interconnect devices. By default, it displays only disk, network, and graphics devices. Use the -v (verbose) option to display all PCI devices.

This utility is intended for debugging purposes; running it on an active system may cause some devices to hang.

Examples:

Here's some sample output from `pci`:

```
Class          = Mass Storage (IDE)
Vendor ID      = 8086, Intel Corporation
Device ID      = 7010h, 82371SB PIIX3 IDE Interface (Triton II)
PCI index      = 0
IO Address     = ffa0h enabled
PCI Int Pin    = NC
Interrupt line = 0
```


Copyright © 2014, QNX Software Systems Limited

pci-bios, pci-bios-v2

Provide support for the PCI BIOS

You must be `root` to start this server.

Syntax:

For `pci-bios`:

```
pci-bios [-b buses] [-B] [-c] [-m] [-v] [-x] [-dbios bios_options]
```

You must run `pci-bios-v2` as `pci-bios`:

```
pci-bios [-b buses] [-B] [-c] [-D] [-M] [-m] [-P addr:size]
         [-v] [-x] [-dbios bios_options]
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

-b *buses*

The number of PCI buses to scan. The default is 10.

-B

Force enumeration of bridges of type “OTHER”.

-c

Ignore class code check.

-D

(`pci-bios-v2` only) Enable Message Signaled Interrupts (MSI) and Extended MSI (MSI-X) for video.

-dbios *bios_options*

BIOS options, which include:

irqlist=*irq1*,*irq2*,...

Copyright © 2014, QNX Software Systems Limited

Pass a list of IRQs to the BIOS. For example:

```
irqlist=5,7,9
```

-M

(`pci-bios-v2` only) Disable MSI and MSI-X (they're enabled by default).

-m

Don't map IRQs; some BIOSs don't support IRQ mapping.

-P ***addr*:*size***

(`pci-bios-v2` only) Specify the address and size of the extended PCIe configuration space.

-v

Verbose output; display detailed information on the console.

-x

Don't remove devices from the PCI bus while enumerating them.

Description:

The `pci-bios` server provides PCI BIOS support. You'll have to provide it in your boot image for systems with a PCI BIOS. Invoke *`seedres`* (p. 1740) before starting `pci-bios`.

This server creates the `/dev/pci` device. Wait for it to appear by specifying the following in the buildfile used by *`mkifs`* (p. 1241):

```
pci-bios
waitfor /dev/pci
```

The `pci-bios-v2` server is similar to `pci-bios`, but supports Message Signaled Interrupts (MSI). You must use `pci-bios-v2` if you're using *`startup-apic`* (p. 1854).

If you're using `pci-bios-v2`, you must name it `pci-bios` in order for the enumerators to work correctly. In your buildfile, add `pci-bios-v2` like this:

```
pci-bios=pci-bios-v2
```

To assign MSI or MSI-X interrupts in a driver, use the `PCI_USE_MSI` or `PCI_USE_MSIX` flag when you call *pci_attach_device()*. For more information, see its entry in the QNX Neutrino *C Library Reference*.

Copyright © 2014, QNX Software Systems Limited

pcnfsd

Unix authenticator for NFS

Syntax:

```
pcnfsd
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `pcnfsd` utility is used for authentication purposes by NFS clients in a PC environment that has no concept of user/password identification (e.g. DOS).

There's no direct check for `root` on the mount. The `nfsd` utility checks only that the requests come from a privileged port (which implies `root` access). The check may be disabled using the -norsvd option in the */etc/exports* (p. 721) file.

If a configuration file is present (*/etc/pcnfsd.conf* (p. 1458)) `pcnfsd` uses the information it contains to replace certain builtin elements.

Copyright © 2014, QNX Software Systems Limited

/etc/pcnfsd.conf

Configuration file for `pcnfsd` *definitions*

Name:

`/etc/pcnfsd.conf`

Description:

The `pcnfsd.conf` file contains a range of acceptable user IDs. The `pcnfsd` utility uses this file to authenticate users trying to log on. The default range is `101-60002`.

The file is in the format:

```
keyword argument
```

where *keyword* and *argument* are as follows:

keyword	*argument*	Meaning
`uidrange`	*value-value*	Accepted range of user IDs
`uidrange`	*value*	Accepted single user ID

Format rules:

- There is a space between *keyword* and *argument*.
- Lines that start with # are comments.
- Blank lines are ignored.

For example:

To authenticate users with a uid between 100 and 200:

```
uidrange 100-200
```

To authenticate a user with a uid of 152:

```
uidrange 152
```

To authenticate several uid ranges:

```
uidrange 5
uidrange 150-200
uidrange 500-600
```


Copyright © 2014, QNX Software Systems Limited

pdebug

Process-level debugger

Syntax:

```
pdebug [-1eflsv] [-n priority_levels] device
```

Runs on:

QNX Neutrino

Options:

-1

("One") Exit pdebug after the debugging session is done.

-e

Echo the debugged program's *stdin* back to the host.

-f

Run pdebug as a foreground process.

-l

("el") By default, pdebug sets the ***LD_BIND_NOW*** to 1 to force all binding to be done immediately instead of lazily. To permit lazy binding, specify the -l option. For more information, see "Optimizing the runtime linker" in the Compiling and Debugging chapter of the QNX Neutrino *Programmer's Guide*.

-n *priority_levels*

Be nice; set the debugged program's priority to be *priority_levels* lower than pdebug's. Doing this can keep pdebug from becoming unresponsive if the debugged process misbehaves (e.g. looping in a tight loop taking lots of CPU time).

-s

Notify the host only of signals caused by faults.

-v

Be verbose.

device

Copyright © 2014, QNX Software Systems Limited

The device to use for the remote debug protocol; one of:

-

Use *stdout* / *stdin*.

***/device_name*[,*baud*]**

Open and use the specified device, such as `/dev/ser1`, optionally setting the baud rate.

number

Accept connections on TCP/IP port *number*.

Description:

This utility provides access to process-level debugging from a remote host, including from the IDE. To use `pdebug`, you need to run *`devc-pty`* (p. 288) on the target machine (i.e. the machine being debugged).

Examples:

For a 57600 baud serial connection on `/dev/ser2`:

```
pdebug /dev/ser2,57600 &
```

For a TCP/IP connection on port 8000:

```
pdebug 8000 &
```


Copyright © 2014, QNX Software Systems Limited

pf

Packet Filter pseudo-device

Name:

`/dev/pf`

Description:

Packet filtering takes place in `io-pkt`. A pseudo-device, `/dev/pf`, lets user processes control the behavior of the packet filter through an *ioctl()* interface. There are commands to enable and disable the filter, load rule sets, add and remove individual rules or state table entries, and retrieve statistics. The most commonly used functions are covered by *`pfctl`* (p. 1513).

If you're using QNX Neutrino 6.4.1 or earlier, you should use *ioctl_socket()* instead of *ioctl()* in your packet-filtering code. With the microkernel message-passing architecture, *ioctl()* calls that have pointers embedded in them need to be handled specially. The *ioctl_socket()* function uses *ioctl()* for functionality that doesn't require special handling.

In QNX Neutrino 6.5.0 and later, *ioctl()* handles embedded pointers, so you don't have to use *ioctl_socket()* instead.

Manipulations such as loading a rule set that involve more than a single *ioctl()* call require a *ticket*, which prevents the occurrence of multiple concurrent manipulations.

Fields of *ioctl()* parameter structures that refer to packet data (such as addresses and ports) are generally expected in network-byte order.

Rules and address tables are contained in *anchors*. When servicing an *ioctl()* request, if the anchor field of the argument structure is empty, `io-pkt` uses the default anchor (i.e. the main rule set) in operations. Anchors are specified by name and may be nested, with components separated by slashes, similar to the way that filesystem hierarchies are laid out. The final component of the anchor path is the anchor under which operations will be performed.

ioctl() interface

The `pf` pseudo-device supports the following *ioctl()* commands, available through `<net/pfvar.h>`:

DIOCSTART

Start the packet filter.

DIOCSTOP

Copyright © 2014, QNX Software Systems Limited

Stop the packet filter.

DIOCSTARTALTQ

Start the ALTQ bandwidth control system (see *altq* in the NetBSD documentation).

DIOCSTOPALTQ

Stop the ALTQ bandwidth control system.

DIOCBEGINADDRS struct pfioc_pooladdr *pp

Clear the buffer address pool and get a ticket for subsequent DIOCADDADDR, DIOCADDRULE, and DIOCCHANGERULE calls. The pfioc_pooladdr structure is defined as follows:

```
struct pfioc_pooladdr {
   u_int32_t               action;
   u_int32_t               ticket;
   u_int32_t               nr;
   u_int32_t               r_num;
   u_int8_t                r_action;
   u_int8_t                r_last;
   u_int8_t                af;
   char                    anchor[MAXPATHLEN];
   struct pf_pooladdr      addr;
};
```

DIOCADDADDR struct pfioc_pooladdr *pp

Add the pool address, *addr* to the buffer address pool to be used in the following DIOCADDRULE or DIOCCHANGERULE call. All other members of the structure are ignored.

DIOCADDRULE struct pfioc_rule *pr

Add the given rule at the end of the inactive rule set. The pfioc_rule structure is defined as follows:

```
struct pfioc_rule {
        u_int32_t       action;
        u_int32_t       ticket;
        u_int32_t       pool_ticket;
        u_int32_t       nr;
        char            anchor[MAXPATHLEN];
        char            anchor_call[MAXPATHLEN];
        struct pf_rule  rule;
};
```

This call requires a *ticket* obtained through a preceding DIOCXBEGIN call and a *pool_ticket* obtained through a DIOCBEGINADDRS call. You must also call DIOCADDADDR if any pool addresses are required.

The optional anchor name indicates the anchor in which to append the rule. The *nr* and *action* members. are ignored.

Copyright © 2014, QNX Software Systems Limited

DIOCADDALTQ struct pfioc_altq *pa

Add an ALTQ discipline or queue. The `pfioc_altq` structure is defined as follows:

```
struct pfioc_altq {
        u_int32_t       action;
        u_int32_t       ticket;
        u_int32_t       nr;
        struct pf_altq  altq;
};
```

DIOCGETRULES struct pfioc_rule *pr

Get a ticket for subsequent `DIOCGETRULE` calls, and the number *nr* of rules in the active rule set.

DIOCGETRULE struct pfioc_rule *pr

Get a rule by its number *nr*, using the ticket obtained through a preceding `DIOCGETRULES` call.

DIOCGETADDRS struct pfioc_pooladdr *pp

Get a ticket for subsequent `DIOCGETADDR` calls and the number *nr* of pool addresses in the rule specified with *r_action*, *r_num*, and *anchor*.

DIOCGETADDR struct pfioc_pooladdr *pp

Get the pool address *addr* by its number *nr* from the rule specified with *r_action*, *r_num*, and *anchor*, using the ticket obtained through a preceding `DIOCGETADDRS` call.

DIOCGETALTQS struct pfioc_altq *pa

Get a ticket for subsequent `DIOCGETALTQ` calls and the number *nr* of queues in the active list.

DIOCGETALTQ struct pfioc_altq *pa

Get the queueing discipline *altq* by its number *nr*, using the ticket obtained through a preceding `DIOCGETALTQS` call.

DIOCGETQSTATS struct pfioc_qstats *pq

Get the statistics for a queue. The `pfioc_qstats` structure is defined as follows:

```
struct pfioc_qstats {
        u_int32_t        ticket;
        u_int32_t        nr;
        void            *buf;
        int nbytes;
        u_int8_t         scheduler;
};
```

Copyright © 2014, QNX Software Systems Limited

This call fills in a pointer to the buffer of statistics *buf*, of length *nbytes*, for the queue specified by *nr*.

DIOCGETRULESETS struct pfioc_ruleset *pr

Get the number *nr* of rule sets (i.e., anchors) directly attached to the anchor named by *path* for use in subsequent DIOCGETRULESET calls. The pfioc_ruleset structure is defined as follows:

```
struct pfioc_ruleset {
        u_int32_t        nr;
        char             path[MAXPATHLEN];
        char             name[PF_ANCHOR_NAME_SIZE];
};
```

Nested anchors, since they aren't directly attached to the given anchor, aren't included. This *ioctl()* command returns EINVAL if the given anchor doesn't exist.

DIOCGETRULESET struct pfioc_ruleset *pr

Get a rule set (i.e., an anchor) name by its number *nr* from the given anchor path, the maximum number of which can be obtained from a preceding DIOCGETRULESETS call. This *ioctl()* command returns EINVAL if the given anchor doesn't exist, or EBUSY if another process is concurrently updating a rule set.

DIOCADDSTATE struct pfioc_state *ps

Add a state entry. The pfioc_state structure is defined as follows:

```
struct pfioc_state {
        u_int32_t        nr;
        struct pf_state  state;
};
```

DIOCGETSTATE struct pfioc_state *ps

Extract the entry with the specified number *nr* from the state table.

DIOCKILLSTATES struct pfioc_state_kill *psk

Remove matching entries from the state table. This *ioctl()* command returns the number of killed states in *psk_af*. The pfioc_state_kill structure is defined as follows:

```
struct pfioc_state_kill {
        sa_family_t              psk_af;
        int                      psk_proto;
        struct pf_rule_addr      psk_src;
        struct pf_rule_addr      psk_dst;
        char                     psk_ifname[IFNAMSIZ];
};
```


Copyright © 2014, QNX Software Systems Limited

DIOCCLRSTATES struct pfioc_state_kill *psk

Clear all states. This command works like DIOCKILLSTATES, but ignores the *psk_af*, *psk_proto*, *psk_src*, and *psk_dst* fields of the pfioc_state_kill structure.

DIOCSETSTATUSIF struct pfioc_if *pi

Specify the interface for which to gather statistics. The pfioc_if structure is defined as follows:

```
struct pfioc_if {
        char             ifname[IFNAMSIZ];
};
```

DIOCGETSTATUS struct pf_status *s

Get the internal packet filter statistics. The pf_status structure is defined as follows:

```
struct pf_status {
        u_int64_t       counters[PFRES_MAX];
        u_int64_t       lcounters[LCNT_MAX];
        u_int64_t       fcounters[FCNT_MAX];
        u_int64_t       scounters[SCNT_MAX];
        u_int64_t       pcounters[2][2][3];
        u_int64_t       bcounters[2][2];
        u_int64_t       stateid;
        u_int32_t       running;
        u_int32_t       states;
        u_int32_t       src_nodes;
        u_int32_t       since;
        u_int32_t       debug;
        u_int32_t       hostid;
        char            ifname[IFNAMSIZ];
};
```

DIOCCLRSTATUS

Clear the internal packet filter statistics.

DIOCNATLOOK struct pfioc_natlook *pnl

Look up a state table entry by source and destination addresses and ports. The pfioc_natlook structure is defined as follows:

```
struct pfioc_natlook {
        struct pf_addr   saddr;
        struct pf_addr   daddr;
        struct pf_addr   rsaddr;
        struct pf_addr   rdaddr;
        u_int16_t        sport;
        u_int16_t        dport;
        u_int16_t        rsport;
        u_int16_t        rdport;
        sa_family_t      af;
        u_int8_t         proto;
        u_int8_t         direction;
};
```

Copyright © 2014, QNX Software Systems Limited

DIOCSETDEBUG u_int32_t *level

Set the debug level to one of PF_DEBUG_NONE, PF_DEBUG_URGENT, PF_DEBUG_MISC, or PF_DEBUG_NOISY.

DIOCGETSTATES struct pfioc_states *ps

Get state table entries. The pfioc_states structure is defined as follows:

```
struct pfioc_states {
        int     ps_len;
        union {
                caddr_t          psu_buf;
                struct pf_state *psu_states;
        } ps_u;
#define ps_buf          ps_u.psu_buf
#define ps_states       ps_u.psu_states
};
```

If *ps_len* is zero, all states are gathered into *pf_states*, and *ps_len* is set to the size they take in memory (i.e. sizeof(struct pf_state) * nr). If *ps_len* is nonzero, as many states that can fit into *ps_len* as possible are gathered, and *ps_len* is updated to the size those rules take in memory.

DIOCCHANGERULE struct pfioc_rule *pcr

Add or remove the rule in the rule set specified by *rule.action*. The type of operation to be performed is indicated by *action*, which can be any of the following:

- PF_CHANGE_NONE
- PF_CHANGE_ADD_HEAD
- PF_CHANGE_ADD_TAIL
- PF_CHANGE_ADD_BEFORE
- PF_CHANGE_ADD_AFTER
- PF_CHANGE_REMOVE
- PF_CHANGE_GET_TICKET

You must set *ticket* to the value obtained with PF_CHANGE_GET_TICKET for all actions except PF_CHANGE_GET_TICKET. You must set *pool_ticket* to the value obtained with the DIOCBEGINADDRS call for all actions except PF_CHANGE_REMOVE and PF_CHANGE_GET_TICKET. The *anchor* indicates which anchor the operation applies to. The *nr* member indicates the rule number against which to apply PF_CHANGE_ADD_BEFORE, PF_CHANGE_ADD_AFTER, or PF_CHANGE_REMOVE actions.

DIOCCHANGEADDR struct pfioc_pooladdr *pca

Add or remove the pool address *addr* from the rule specified by *r_action*, *r_num*, and *anchor*.

Copyright © 2014, QNX Software Systems Limited

DIOCSETTIMEOUT struct pfioc_tm *pt

Set the state timeout of *timeout* to *seconds*. The `pfioc_tm` structure is defined as follows:

```
struct pfioc_tm {
        int timeout;
        int seconds;
};
```

The old value is placed into *seconds*. For the possible values of *timeout*, see the `PFTM_*` values in `<net/pfvar.h>`.

DIOCGETTIMEOUT struct pfioc_tm *pt

Get the state timeout of *timeout*. The value is placed into the *seconds* field.

DIOCCLRRULECTRS

Clear per-rule statistics.

DIOCSETLIMIT struct pfioc_limit *pl

Set the hard limits on the memory pools used by the packet filter. The `pfioc_limit` structure is defined as follows:

```
struct pfioc_limit {
        int        index;
        unsigned   limit;
};

enum { PF_LIMIT_STATES, PF_LIMIT_SRC_NODES,
       PF_LIMIT_FRAGS };
```

DIOCGETLIMIT struct pfioc_limit *pl

Get the hard limit for the memory pool indicated by *index*.

DIOCRCLRTABLES struct pfioc_table *io

Clear all tables. All the *ioctl()* commands that manipulate radix tables use the same structure described below. On exit from the `DIOCRCLRTABLES` command, *pfrio_ndel* contains the number of tables deleted. The `pfioc_table` structure is defined as follows:

```
struct pfioc_table {
        struct pfr_table         pfrio_table;
        void                    *pfrio_buffer;
        int                      pfrio_esize;
        int                      pfrio_size;
        int                      pfrio_size2;
        int                      pfrio_nadd;
        int                      pfrio_ndel;
        int                      pfrio_nchange;
        int                      pfrio_flags;
        u_int32_t                pfrio_ticket;
};

#define pfrio_exists    pfrio_nadd
#define pfrio_nzero     pfrio_nadd
```

Copyright © 2014, QNX Software Systems Limited

```
#define pfrio_nmatch    pfrio_nadd
#define pfrio_naddr     pfrio_size2
#define pfrio_setflag   pfrio_size2
#define pfrio_clrflag   pfrio_nadd
```

DIOCRADDTABLES struct pfioc_table *io

Create one or more tables. On entry, *pfrio_buffer[pfrio_size]* contains a table of `pfr_table` structures. On exit, *pfrio_nadd* contains the number of tables effectively created. The `pfr_table` structure is defined as follows:

```
struct pfr_table {
        char            pfrt_anchor[MAXPATHLEN];
        char            pfrt_name[PF_TABLE_NAME_SIZE];
        u_int32_t       pfrt_flags;
        u_int8_t        pfrt_fback;
};
```

DIOCRDELTABLES struct pfioc_table *io

Delete one or more tables. On entry, *pfrio_buffer[pfrio_size]* contains a table of `pfr_table` structures. On exit, *pfrio_nadd* contains the number of tables effectively deleted.

DIOCRGETTABLES struct pfioc_table *io

Get a list of all tables. On entry, *pfrio_buffer[pfrio_size]* contains a valid writeable buffer for `pfr_table` structures. On exit, *pfrio_size* contains the number of tables written into the buffer. If the buffer is too small, `io-pkt` doesn't store anything, but just returns the required buffer size, without error.

DIOCRGETTSTATS struct pfioc_table *io

This call is like `DIOCRGETTABLES`, but is used to get an array of `pfr_tstats` structures. The `pfr_tstats` structure is defined as follows:

```
struct pfr_tstats {
        struct pfr_table pfrts_t;
        u_int64_t        pfrts_packets
                             [PFR_DIR_MAX][PFR_OP_TABLE_MAX];
        u_int64_t        pfrts_bytes
                             [PFR_DIR_MAX][PFR_OP_TABLE_MAX];
        u_int64_t        pfrts_match;
        u_int64_t        pfrts_nomatch;
        long             pfrts_tzero;
        int              pfrts_cnt;
        int              pfrts_refcnt[PFR_REFCNT_MAX];
};
#define pfrts_name       pfrts_t.pfrt_name
#define pfrts_flags      pfrts_t.pfrt_flags
```

DIOCRCLRTSTATS struct pfioc_table *io

Copyright © 2014, QNX Software Systems Limited

Clear the statistics of one or more tables. On entry, *pfrio_buffer[pfrio_size]* contains a table of `pfr_table` structures. On exit, *pfrio_nzero* contains the number of tables effectively cleared.

DIOCRCLRADDRS struct pfioc_table *io

Clear all addresses in a table. On entry, *pfrio_table* contains the table to clear. On exit, *pfrio_ndel* contains the number of addresses removed.

DIOCRADDADDRS struct pfioc_table *io

Add one or more addresses to a table. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains the list of `pfr_addr` structures to add. On exit, *pfrio_nadd* contains the number of addresses effectively added.

The `pfr_addr` structure is defined as follows:

```
struct pfr_addr {
        union {
                struct in_addr   _pfra_ip4addr;
                struct in6_addr  _pfra_ip6addr;
        }   pfra_u;
        u_int8_t         pfra_af;
        u_int8_t         pfra_net;
        u_int8_t         pfra_not;
        u_int8_t         pfra_fback;
};
#define pfra_ip4addr    pfra_u._pfra_ip4addr
#define pfra_ip6addr    pfra_u._pfra_ip6addr
```

DIOCRDELADDRS struct pfioc_table *io

Delete one or more addresses from a table. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains the list of `pfr_addr` structures to delete. On exit, *pfrio_ndel* contains the number of addresses effectively deleted.

DIOCRSETADDRS struct pfioc_table *io

Replace the content of a table by a new address list. This is the most complicated command, which uses all the structure members.

On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains the new list of `pfr_addr` structures. Additionally, if *pfrio_size2* is nonzero, *pfrio_buffer[pfrio_size..pfrio_size2]* must be a writeable buffer, into which `io-pkt` can copy the addresses that have been deleted during the replace operation.

On exit, *pfrio_ndel*, *pfrio_nadd*, and *pfrio_nchange* contain the number of addresses deleted, added, and changed by `io-pkt`. If *pfrio_size2* was set on entry, *pfrio_size2* points to the size of the buffer used, exactly as for DIOCRGETADDRS.

Copyright © 2014, QNX Software Systems Limited

DIOCRGETADDRS struct pfioc_table *io

Get all the addresses of a table. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains a valid writeable buffer for `pfr_addr` structures. On exit, *pfrio_size* contains the number of addresses written into the buffer. If the buffer is too small, `io-pkt` doesn't store anything, but just returns the required buffer size, without returning an error.

DIOCRGETASTATS struct pfioc_table *io

This call is like DIOCRGETADDRS, but is used to get an array of `pfr_astats` structures:

```
struct pfr_astats {
        struct pfr_addr  pfras_a;
        u_int64_t        pfras_packets
                [PFR_DIR_MAX][PFR_OP_ADDR_MAX];
        u_int64_t        pfras_bytes
                [PFR_DIR_MAX][PFR_OP_ADDR_MAX];
        long             pfras_tzero;
};
```

DIOCRCLRASTATS struct pfioc_table *io

Clear the statistics of one or more addresses. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains a table of `pfr_addr` structures to clear. On exit, *pfrio_nzero* contains the number of addresses effectively cleared.

DIOCRTSTADDRS struct pfioc_table *io

Test if the given addresses match a table. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains a table of `pfr_addr` structures to test. On exit, `io-pkt` updates the *pfr_addr* table by setting the *pfra_fback* member appropriately.

DIOCRSETTFLAGS struct pfioc_table *io

Change the PFR_TFLAG_CONST or PFR_TFLAG_PERSIST flags of a table. On entry, *pfrio_buffer[pfrio_size]* contains a table of `pfr_table` structures, and *pfrio_setflag* contains the flags to add, while *pfrio_clrflag* contains the flags to remove. On exit, *pfrio_nchange* and *pfrio_ndel* contain the number of tables altered or deleted by `io-pkt`.

You can delete tables if you remove the PFR_TFLAG_PERSIST flag of an unreferenced table.

DIOCRINADEFINE struct pfioc_table *io

Copyright © 2014, QNX Software Systems Limited

Define a table in the inactive set. On entry, *pfrio_table* contains the table ID, and *pfrio_buffer[pfrio_size]* contains the list of `pfr_addr` structures to put in the table. A valid ticket must also be supplied to *pfrio_ticket*. On exit, *pfrio_nadd* contains 0 if the table was already defined in the inactive list, or 1 if a new table has been created. The *pfrio_naddr* member contains the number of addresses effectively put in the table.

DIOCXBEGIN struct pfioc_trans *io

Clear all the inactive rule sets specified in the *pfioc_trans_e* array. The `pfioc_trans` structure is defined as follows:

```
struct pfioc_trans {
        int  size;  /* number of elements */
        int  esize; /* size of each element in bytes */
        struct pfioc_trans_e {
           int             rs_num;
           char            anchor[MAXPATHLEN];
           u_int32_t       ticket;
        } *array;
};
```

For each rule set, a ticket is returned for subsequent "add rule" *ioctl()* commands, as well as for the `DIOCXCOMMIT` and `DIOCXROLLBACK` calls.

Rule set types, identified by *rs_num*, include the following:

- `PF_RULESET_SCRUB` — scrub (packet normalization) rules.
- `PF_RULESET_FILTER` — filter rules.
- `PF_RULESET_NAT` — NAT (Network Address Translation) rules.
- `PF_RULESET_BINAT` — bidirectional NAT rules.
- `PF_RULESET_RDR` — redirect rules.
- `PF_RULESET_ALTQ` — ALTQ disciplines.
- `PF_RULESET_TABLE` — address tables.

DIOCXCOMMIT struct pfioc_trans *io

Atomically switch a vector of inactive rule sets to the active rule sets. This call is implemented as a standard two-phase commit, which either fails for all rule sets, or completely succeeds. All tickets need to be valid. This *ioctl()* command returns `EBUSY` if another process is concurrently updating some of the same rule sets.

DIOCXROLLBACK struct pfioc_trans *io

Clean up `io-pkt` by undoing all changes that have taken place on the inactive rule sets since the last `DIOCXBEGIN`. `DIOCXROLLBACK` silently ignores rule sets for which the ticket is invalid.

DIOCSETHOSTID u_int32_t *hostid

Copyright © 2014, QNX Software Systems Limited

Set the host ID, which is used by `pfsync` to identify which host created state table entries.

DIOCOSFPFLUSH

Flush the passive OS fingerprint table.

DIOCOSFPADD struct pf_osfp_ioctl *io

Add a passive OS fingerprint to the table. The `pf_osfp_ioctl` structure is defined as follows:

```
struct pf_osfp_ioctl {
   struct pf_osfp_entry {
      SLIST_ENTRY(pf_osfp_entry) fp_entry;
      pf_osfp_t   fp_os;
      char        fp_class_nm[PF_OSFP_LEN];
      char        fp_version_nm[PF_OSFP_LEN];
      char        fp_subtype_nm[PF_OSFP_LEN];
   } fp_os;
   pf_tcpopts_t            fp_tcpopts;
   u_int16_t               fp_wsize;
   u_int16_t               fp_psize;
   u_int16_t               fp_mss;
   u_int16_t               fp_flags;
   u_int8_t                fp_optcnt;
   u_int8_t                fp_wscale;
   u_int8_t                fp_ttl;
   int                     fp_getnum;
};
```

Set *fp_os.fp_os* to the packed fingerprint, *fp_os.fp_class_nm* to the name of the class (Linux, Windows, etc.), *fp_os.fp_version_nm* to the name of the version (NT, 95, 98), and *fp_os.fp_subtype_nm* to the name of the subtype or patch level. The members *fp_mss*, *fp_wsize*, *fp_psize*, *fp_ttl*, *fp_optcnt*, and *fp_wscale* are set to the TCP MSS, the TCP window size, the IP length, the IP TTL, the number of TCP options, and the TCP window scaling constant of the TCP SYN packet, respectively.

The *fp_flags* member is filled according to the `PF_OSFP_*` definition in `<net/pfvar.h>`. The *fp_tcpopts* member contains packed TCP options. Each option uses `PF_OSFP_TCPOPT_BITS` bits in the packed value. Options include any of `PF_OSFP_TCPOPT_NOP`, `PF_OSFP_TCPOPT_SACK`, `PF_OSFP_TCPOPT_WSCALE`, `PF_OSFP_TCPOPT_MSS`, or `PF_OSFP_TCPOPT_TS`.

This *ioctl()* command doesn't use the *fp_getnum* member.

You must zero the structure's slack space for correct operation; *memset()* the whole structure to zero before filling and sending it to `io-pkt`.

DIOCOSFPGET struct pf_osfp_ioctl *io

Copyright © 2014, QNX Software Systems Limited

Get the passive OS fingerprint number *fp_getnum* from `io-pkt`'s fingerprint list. The rest of the structure members comes back filled. Get the whole list by repeatedly incrementing the *fp_getnum* number until *ioctl()* gives an error of EBUSY.

DIOCGETSRCNODES struct pfioc_src_nodes *psn

Get the list of source nodes kept by sticky addresses and source tracking. The `pfioc_src_nodes` structure is defined as follows:

```
struct pfioc_src_nodes {
   int     psn_len;
   union {
           caddr_t         psu_buf;
           struct pf_src_node      *psu_src_nodes;
   } psn_u;
#define psn_buf         psn_u.psu_buf
#define psn_src_nodes   psn_u.psu_src_nodes
};
```

You must call *ioctl()* once with *psn_len* set to 0. If *ioctl()* returns without error, *psn_len* is set to the size of the buffer required to hold all the pf_src_node structures held in the table. You should then allocate a buffer of this size and place a pointer to this buffer in *psn_buf*. You must then call *ioctl()* again to fill this buffer with the actual source node data. After that call, *psn_len* is set to the length of the buffer actually used.

DIOCCLRSRCNODES

Clear the tree of source-tracking nodes.

DIOCIGETIFACES struct pfioc_iface *io

Get a list of interfaces and interface drivers known to `pf`. All the *ioctl()* commands that manipulate interfaces use the structure described below:

```
struct pfioc_iface {
        char                    pfiio_name[IFNAMSIZ];
        void                   *pfiio_buffer;
        int                     pfiio_esize;
        int                     pfiio_size;
        int                     pfiio_nzero;
        int                     pfiio_flags;
};

#define PFI_FLAG_GROUP     0x0001  /* gets groups of interfaces */
#define PFI_FLAG_INSTANCE  0x0002  /* gets single interfaces */
#define PFI_FLAG_ALLMASK   0x0003
```

If it isn't empty, you can use *pfiio_name* to restrict the search to a specific interface or driver. The *pfiio_buffer[pfiio_size]* member is the user-supplied buffer for returning the data. On entry, *pfiio_size* represents the number of `pfi_if` entries that can fit into the buffer. The `io-pkt` manager replaces this value with the real number of entries it wants to return.

Copyright © 2014, QNX Software Systems Limited

You should set *pfiio_esize* to `sizeof(struct pfi_if)`. You should set *pfiio_flags* to `PFI_FLAG_GROUP`, `PFI_FLAG_INSTANCE`, or both, to tell `io-pkt` to return a group of interfaces (drivers, such as `fxp`), real interface instances (e.g. `fxp1`), or both. The data is returned in the `pfi_if` structure described below:

```
struct pfi_if {
        char        pfif_name[IFNAMSIZ];
        u_int64_t   pfif_packets[2][2][2];
        u_int64_t   pfif_bytes[2][2][2];
        u_int64_t   pfif_addcnt;
        u_int64_t   pfif_delcnt;
        long        pfif_tzero;
        int         pfif_states;
        int         pfif_rules;
        int         pfif_flags;
};

#define PFI_IFLAG_GROUP    0x0001  /* group of interfaces */
#define PFI_IFLAG_INSTANCE 0x0002  /* single instance */
#define PFI_IFLAG_CLONABLE 0x0010  /* clonable group */
#define PFI_IFLAG_DYNAMIC  0x0020  /* dynamic group */
#define PFI_IFLAG_ATTACHED 0x0040  /* interface attached */
```

`DIOCICLRISTATS struct pfioc_iface *io`

Clear the statistics counters of one or more interfaces. You can use *pfiio_name* and *pfiio_flags* to select which interfaces need to be cleared. The filtering process is the same as for `DIOCIGETIFACES`. The *pfiio_nzero* member is set by `io-pkt` to the number of interfaces and drivers that have been cleared.

`DIOCSETIFFLAG struct pfioc_iface *io`

Set the user-setable flags of the `pf` internal interface description:

- `PFI_IFLAG_SKIP` — skip the interface.

The filtering process is the same as for `DIOCIGETIFACES`.

`DIOCCLRIFFLAG struct pfioc_iface *io`

Similar to `DIOCSETIFFLAG`, but clear the flags.

Examples:

The following example demonstrates how to use the `DIOCNATLOOK` command to find the internal host/port of a NATed connection:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <sys/fcntl.h>
#include <net/if.h>
#include <netinet/in.h>
#include <net/pfvar.h>
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
```


Copyright © 2014, QNX Software Systems Limited

```
u_int32_t
read_address(const char *s)
{
        int a, b, c, d;

        sscanf(s, "%i.%i.%i.%i", &a, &b, &c, &d);
        return htonl(a << 24 | b << 16 | c << 8 | d);
}

void
print_address(u_int32_t a)
{
        a = ntohl(a);
        printf("%d.%d.%d.%d", a >> 24 & 255, a >> 16 & 255,
            a >> 8 & 255, a & 255);
}

int
main(int argc, char *argv[])
{
        struct pfioc_natlook nl;
        int dev;

        if (argc != 5) {
           printf("%s <gwy addr> <gwy port> <ext addr> <ext port>\n",
                  argv[0]);
           return 1;
        }

        dev = open("/dev/pf", O_RDWR);
        if (dev == -1)
                err(1, "open(\"/dev/pf\") failed");

        memset(&nl, 0, sizeof(struct pfioc_natlook));
        nl.saddr.v4.s_addr      = read_address(argv[1]);
        nl.sport                = htons(atoi(argv[2]));
        nl.daddr.v4.s_addr      = read_address(argv[3]);
        nl.dport                = htons(atoi(argv[4]));
        nl.af                   = AF_INET;
        nl.proto                = IPPROTO_TCP;
        nl.direction            = PF_IN;

        if (ioctl(dev, DIOCNATLOOK, &nl))
                err(1, "DIOCNATLOOK");

        printf("internal host ");
        print_address(nl.rsaddr.v4.s_addr);
        printf(":%u\n", ntohs(nl.rsport));
        return 0;
}
```

Caveats:

The following functionality is missing from `pf` in this version of NetBSD:

- The `pfsync` protocol isn't supported.
- The `group` keyword isn't supported.

Copyright © 2014, QNX Software Systems Limited

/etc/pf.conf

Packet filter configuration file

Name:

`/etc/pf.conf`

Description:

The *pf* (p. 1461) packet filter modifies, drops or passes packets according to rules or definitions specified in `pf.conf`. The default location of this file is `/etc/pf.conf`.

Statement order

The `pf.conf` file can include the following types of statements:

Macros

User-defined variables may be defined and used later, simplifying the configuration file. Macros must be defined before they can be referenced in `pf.conf`.

Tables

Tables provide a mechanism for increasing the performance and flexibility of rules with large numbers of source or destination addresses.

Options

Options tune the behavior of the packet filtering engine.

Traffic Normalization (e.g. scrub)

Traffic normalization protects internal machines against inconsistencies in Internet protocols and implementations.

Queueing

Queueing provides rule-based bandwidth control.

Translation (Various forms of NAT)

Translation rules specify how addresses are to be mapped or redirected to other addresses.

Packet Filtering

Stateful and stateless packet filtering provides rule-based blocking or passing of packets.

Copyright © 2014, QNX Software Systems Limited

With the exception of macros and tables, the types of statements should be grouped and appear in `pf.conf` in the order shown above, as this matches the operation of the underlying packet filtering engine. By default, *`pfctl`* (p. 1513) enforces this order (see `set require-order` below).

Macros

Much as for `cpp` or `m4`, you can define macros that will later be expanded in context. Macro names must start with a letter, and may contain letters, digits and underscores. Macro names may not be reserved words (for example, `pass`, `in`, `out`). Macros aren't expanded inside quotes.

For example:

```
ext_if = "kue0"
all_ifs = "{" $ext_if lo0 "}"
pass out on $ext_if from any to any keep state
pass in  on $ext_if proto tcp from any to any port 25 keep state
```

Tables

Tables are named structures that can hold a collection of addresses and networks. Lookups against tables in `pf` are relatively fast, making a single rule with tables much more efficient, in terms of processor usage and memory consumption, than a large number of rules that differ only in IP address (either created explicitly or automatically by rule expansion).

You can use tables as the source or destination of filter rules, scrub rules or translation rules such as `nat` or `rdr` (see below for details on the various rule types) You can also use tables for the redirect address of `nat` and `rdr` rules and in the routing options of filter rules, but only for round-robin pools.

You can define tables with any of the following `pfctl` mechanisms. As with macros, reserved words may not be used as table names.

Manually

Persistent tables can be manually created with the add or replace option of `pfctl`, before or after the rule set has been loaded.

Via `pf.conf`

Table definitions can be placed directly in this file, and loaded at the same time as other rules are loaded, atomically. Table definitions inside `pf.conf` use the table statement, and are especially useful to define non-persistent tables. The contents of a pre-existing table defined without a list of addresses to initialize it isn't altered when `pf.conf` is loaded. A table initialized with the empty list, `{ }`, is cleared on loading.

Tables may be defined with the following attributes:

`persist`

Copyright © 2014, QNX Software Systems Limited

Force `io-pkt` to keep the table even when no rules refer to it. If the flag isn't set, `io-pkt` automatically removes the table when the last rule referring to it is flushed.

`const`

Prevent the user from altering the contents of the table once it's been created. Without this flag, you can use `pfctl` to add or remove addresses from the table at any time, even when running with `securelevel = 2`.

For example:

```
table <private> const { 10/8, 172.16/12, 192.168/16 }
table <badhosts> persist
block on fxp0 from { <private>, <badhosts> } to any
```

creates a table called `private`, to hold RFC 1918 private network blocks, and a table called `badhosts`, which is initially empty. A filter rule is set up to block all traffic coming from addresses listed in either table.

The `private` table can't have its contents changed, and the `badhosts` table will exist even when no active filter rules refer to it. Addresses may later be added to the `badhosts` table, so that traffic from these hosts can be blocked by using:

```
# pfctl -t badhosts -Tadd 204.92.77.111
```

A table can also be initialized with an address list specified in one or more external files, using the following syntax:

```
table <spam> persist file "/etc/spammers" file "/etc/openrelays"
block on fxp0 from <spam> to any
```

The files `/etc/spammers` and `/etc/openrelays` list IP addresses, one per line. Any lines beginning with a # are treated as comments and ignored. In addition to being specified by IP address, hosts may also be specified by their hostname. When the resolver is called to add a hostname to a table, all resulting IPv4 and IPv6 addresses are placed into the table. IP addresses can also be entered in a table by specifying a valid interface name or the self keyword, in which case all addresses assigned to the interface(s) are added to the table.

Options

You can tune `pf` for various situations by using the `set` command:

`set timeout` ***arguments***

Set the timeout specified by the arguments:

- `interval` — the interval between the purging of expired states and fragments.
- `frag` — the number of seconds before an unassembled fragment is expired.

Copyright © 2014, QNX Software Systems Limited

- `src.track` — the length of time to retain a source-tracking entry after the last state expires.

When a packet matches a stateful connection, the seconds to live for the connection is updated to that of the `proto.modifier` that corresponds to the connection state. Each packet that matches this state resets the TTL. Tuning these values may improve the performance of the firewall, at the risk of dropping valid idle connections.

- `tcp.first` — the state after the first packet.
- `tcp.opening` — the state before the destination host ever sends a packet.
- `tcp.established` — the fully established state.
- `tcp.closing` — the state after the first FIN has been sent.
- `tcp.finwait` — the state after both FINs have been exchanged and the connection is closed. Some hosts (notably web servers on Solaris) send TCP packets even after closing the connection. Increasing the value of `tcp.finwait` (and possibly that of `tcp.closing`) can prevent blocking of such packets.
- `tcp.closed` — the state after one endpoint sends an RST.

ICMP and UDP are handled in a fashion similar to TCP, but with a much more limited set of states:

- `udp.first` — the state after the first packet.
- `udp.single` — the state if the source host sends more than one packet but the destination host has never sent one back.
- `udp.multiple` — the state if both hosts have sent packets.
- `icmp.first` — the state after the first packet.
- `icmp.error` — the state after an ICMP error came back in response to an ICMP packet.

Other protocols are handled similarly to UDP:

- `other.first`
- `other.single`
- `other.multiple`

Timeout values can be reduced adaptively as the number of state table entries grows.

- `adaptive.start` — when the number of state entries exceeds this value, adaptive scaling begins. All timeout values are scaled linearly with factor (`adaptive.end` – number of states) / (`adaptive.end` – `adaptive.start`).

Copyright © 2014, QNX Software Systems Limited

- `adaptive.end` — when reaching this number of state entries, all timeout values become zero, effectively purging all state entries immediately. This value is used to define the scale factor; it shouldn't actually be reached (set a lower state limit, see below).

You can define these values both globally and for each rule. When used on a per-rule basis, the values relate to the number of states created by the rule, otherwise to the total number of states. For example:

```
set timeout tcp.first 120
set timeout tcp.established 86400
set timeout { adaptive.start 6000, adaptive.end 12000 }
set limit states 10000
```

With 9000 state table entries, the timeout values are scaled to 50% (`tcp.first` 60, `tcp.established` 43200).

`set loginterface`

Enable the collection of packet and byte count statistics for the given interface. To view these statistics, use:

```
pfctl -s info
```

In this example, `pf` collects statistics on the interface named `dc0`:

```
set loginterface dc0
```

You can disable the log interface by using:

```
set loginterface none
```

`set limit`

Set hard limits on the memory pools used by the packet filter. For example:

```
set limit states 20000
```

sets the maximum number of entries in the memory pool used by state table entries (generated by keep state rules) to 20000. This command:

```
set limit frags 20000
```

sets the maximum number of entries in the memory pool used for fragment reassembly (generated by scrub rules) to 20000. This command:

```
set limit src-nodes 2000
```

sets the maximum number of entries in the memory pool used for tracking source IP addresses (generated by the sticky-address and source-track options) to 2000.

Copyright © 2014, QNX Software Systems Limited

You can combine them:

```
set limit { states 20000, frags 20000, src-nodes 2000 }
```

set optimization

Optimize the engine for one of the following network environments:

- `normal` — a normal network environment. Suitable for almost all networks.
- `high-latency` — a high-latency environment (such as a satellite connection).
- `satellite` — alias for `high-latency`.
- `aggressive` — aggressively expire connections. This can greatly reduce the memory usage of the firewall, at the cost of dropping idle connections early.
- `conservative` — extremely conservative settings. Avoid dropping legitimate connections at the expense of greater memory use (possibly much greater on a busy network) and slightly increased processor utilization.

For example:

```
set optimization aggressive
```

set block-policy

Set the default behavior for the packet block action:

- `drop` — the packet is silently dropped.
- `return` — a TCP RST is returned for blocked TCP packets, an ICMP UNREACHABLE is returned for blocked UDP packets, and all other packets are silently dropped.

For example:

```
set block-policy return
```

set state-policy

Set the default behavior for states:

- `if-bound` — states are bound to interface.
- `group-bound` — states are bound to interface group (e.g. ppp).
- `floating` — states can match packets on any interfaces (the default).

Copyright © 2014, QNX Software Systems Limited

For example:

```
set state-policy if-bound
```

set require-order

By default, `pfctl` enforces an ordering of the statement types in the rule set to: options, normalization, queueing, translation, and filtering. Setting this option to `no` disables this enforcement.

There may be non-trivial and non-obvious implications to an out-of-order rule set. Consider carefully before disabling the order enforcement.

set fingerprints

Load fingerprints of known operating systems from the given file name. By default, fingerprints of known operating systems are automatically loaded from `/etc/pf.os`, but you can override the path with this option. For example:

```
set fingerprints "/etc/pf.os.devel"
```

Setting this option may leave a small period of time where the fingerprints referenced by the currently active rule set are inconsistent until the new rule set finishes loading.

set skip on *ifspec*

List the interfaces for which packets shouldn't be filtered. Packets passing in or out on such interfaces are passed as if `pf` were disabled, i.e. `pf` doesn't process them in any way. This can be useful on loopback and other virtual interfaces, when packet filtering isn't desired and can have unexpected effects. For example:

```
set skip on lo0
```

set debug

Set the debug level to one of the following:

- `none` — don't generate debug messages.
- `urgent` — generate debug messages only for serious errors.
- `misc` — generate debug messages for various errors.
- `loud` — generate debug messages for common conditions.

Copyright © 2014, QNX Software Systems Limited

set reassemble yes|no

Using scrub rules, fragments can be reassembled by normalization. In this case, fragments are buffered until they form a complete packet, and only the completed packet is passed on to the filter. The advantage is that filter rules have to deal only with complete packets, and can ignore fragments. The drawback of caching fragments is the additional memory cost. This will set whether reassembly occurs during normalization. You will still need to enable the traffic normalization via the `scrub` directive.

Traffic normalization

Traffic normalization is used to sanitize packet content in such a way that there are no ambiguities in packet interpretation on the receiving side. The normalizer does IP fragment reassembly (see `set reassemble` to enable reassembly) to prevent attacks that confuse intrusion detection systems by sending overlapping IP fragments. Packet normalization is invoked with the `scrub` directive, which has the following options:

no-df

Clear the `dont-fragment` bit from a matching IP packet. Some operating systems are known to generate fragmented packets with the `dont-fragment` bit set. This is particularly true with NFS. Scrub will drop such fragmented `dont-fragment` packets unless you specify `no-df`.

Unfortunately some operating systems also generate their `dont-fragment` packets with a IP identification field of zero. Clearing the `dont-fragment` bit on packets with a zero IP ID may cause deleterious results if an upstream router later fragments the packet. Using the `random-id` modifier (see below) is recommended in combination with the `no-df` modifier to ensure unique IP identifiers.

min-ttl ***number***

Enforce a minimum TTL for matching IP packets.

max-mss ***number***

Enforce a maximum MSS for matching TCP packets.

random-id

Replace the IP identification field with random values to compensate for predictable values generated by many hosts. This option applies only to packets that aren't fragmented after the optional fragment reassembly.

reassemble tcp

Copyright © 2014, QNX Software Systems Limited

Statefully normalize TCP connections. The scrub `reassemble tcp` rules may not have the direction (in/out) specified. The `reassemble tcp` performs the following normalizations:

- `ttl` — neither side of the connection is allowed to reduce their IP TTL. An attacker may send a packet such that it reaches the firewall, affects the firewall state, and expires before reaching the destination host. The `reassemble tcp` command raises the TTL of all packets back up to the highest value seen on the connection.
- `timestamp modulation` — modern TCP stacks send a timestamp on every TCP packet and echo the other endpoint's timestamp back to them. Many operating systems will merely start the timestamp at zero when first booted, and increment it several times a second. The uptime of the host can be deduced by reading the timestamp and multiplying it by a constant. Also observing several different timestamps can be used to count hosts behind a NAT device. And spoofing TCP packets into a connection requires knowing or guessing valid timestamps. Timestamps merely need to be monotonically increasing and not derived off a guessable base time. The `reassemble tcp` command will cause scrub to modulate the TCP timestamps with a random number.
- `extended PAWS checks` — there's a problem with TCP on long fat pipes, in that a packet might get delayed for longer than it takes the connection to wrap its 32-bit sequence space. In such an occurrence, the old packet would be indistinguishable from a new packet and would be accepted as such. The solution to this is called PAWS: Protection Against Wrapped Sequence numbers. It protects against it by making sure the timestamp on each packet doesn't go backwards. The `reassemble tcp` command also makes sure the timestamp on the packet doesn't go forward more than the RFC allows. By doing this, `pf` artificially extends the security of TCP sequence numbers by 10 to 18 bits when the host uses appropriately randomized timestamps, since a blind attacker would have to guess the timestamp as well.

For example:

```
scrub in on $ext_if all
```

The `no` option prefixed to a scrub rule causes matching packets to remain unscrubbed, much in the same way as drop quick works in the packet filter (see below). This mechanism should be used when it is necessary to exclude specific packets from broader scrub rules.

Queueing

Copyright © 2014, QNX Software Systems Limited

Packets can be assigned to queues for the purpose of bandwidth control. At least two declarations are required to configure queues, and later any packet filtering rule can reference the defined queues by name. During the filtering component of `pf.conf`, the last referenced queue name is where any packets from pass rules will be queued, while for block rules it specifies where any resulting ICMP or TCP RST packets should be queued. The scheduler defines the algorithm used to decide which packets get delayed, dropped, or sent out immediately.

The currently supported schedulers are:

`cbq`

Class Based Queueing. Queues attached to an interface build a tree, and thus each queue can have further child queues. Each queue can have a priority and a bandwidth assigned. Priority mainly controls the time packets take to get sent out, while bandwidth has primarily effects on throughput.

The `cbq` scheduler achieves both partitioning and sharing of link bandwidth by hierarchically structured classes. Each class has its own queue and is assigned its share of bandwidth. A child class can borrow bandwidth from its parent class, as long as excess bandwidth is available (see the option `borrow`, below).

`priq`

Priority Queueing. Queues are flat attached to the interface, and thus queues can't have further child queues. Each queue has a unique priority assigned, ranging from 0 to 15. Packets in the queue with the highest priority are processed first.

`hfsc`

Hierarchical Fair Service Curve. Queues attached to an interface build a tree, and thus each queue can have further child queues. Each queue can have a priority and a bandwidth assigned. Priority mainly controls the time packets take to get sent out, while bandwidth has primarily effects on throughput.

The `hfsc` scheduler supports both link-sharing and guaranteed realtime services. It employs a service-curve-based QoS model, and its unique feature is an ability to decouple delay and bandwidth allocation.

The interfaces on which queueing should be activated are declared using the `altq on` declaration, which has the following keywords:

interface

Enable queueing on the named interface.

Copyright © 2014, QNX Software Systems Limited

scheduler

Specify which queueing scheduler to use. Currently supported values are:

- `cbq` — Class Based Queueing
- `priq` — Priority Queueing
- `hfsc` — Hierarchical Fair Service Curve

`bandwidth` ***bw***

The maximum bit rate for all queues on an interface. You can specify the value as an absolute value or as a percentage of the interface bandwidth. When using an absolute value, you can use the suffixes `b`, `Kb`, `Mb`, and `Gb` to represent bits, kilobits, megabits, and gigabits per second, respectively. The value mustn't exceed the interface bandwidth. If you don't specify `bandwidth`, the interface bandwidth is used.

`qlimit` ***limit***

The maximum number of packets held in the queue. The default is 50.

`tbrsize` ***size***

Adjust the size, in bytes, of the token bucket regulator. If not specified, heuristics based on the interface bandwidth are used to determine the size.

`queue` ***list***

Define a list of subqueues to create on an interface.

In the following example, the interface `dc0` should queue up to 5 Mbit/s in four second-level queues using Class Based Queueing. Those four queues will be shown in a later example:

```
altq on dc0 cbq bandwidth 5Mb queue { std, http, mail, ssh }
```

Once interfaces are activated for queueing using the `altq` directive, you can define a sequence of queue directives. The name associated with a queue must match a queue defined in the `altq` directive (e.g. `mail`), or — except for the `priq` scheduler — in a parent queue declaration. You can use the following keywords:

- `on` *interface* — the interface the queue operates on. If not given, the queue operates on all matching interfaces.
- `bandwidth` *bw* — the maximum bit rate to be processed by the queue. This value mustn't exceed the value of the parent queue, and you can specify it as an absolute value or a percentage of the parent queue's bandwidth. If you don't specify this keyword, the bandwidth defaults to

Copyright © 2014, QNX Software Systems Limited

100% of the parent queue's bandwidth. The `priq` scheduler doesn't support bandwidth specification.

- `priority` *level* — between queues, a priority level can be set. For `cbq` and `hfsc`, the range is 0 to 7; for `priq`, the range is 0 to 15. The default for all is 1. PRIQ queues with a higher priority are always served first. CBQ and HFSC queues with a higher priority are preferred in the case of overload.
- `qlimit` *limit* — the maximum number of packets held in the queue. The default is 50.

The scheduler can get additional parameters with *scheduler*(*parameters*). The parameters are as follows:

- `default` — packets not matched by another queue are assigned to this one. Exactly one default queue is required.
- `red` — enable RED (Random Early Detection) on this queue. RED drops packets with a probability proportional to the average queue length.
- `rio` — enable RIO on this queue. RIO is RED with IN/OUT, so running RED two times more than RIO would achieve the same effect. RIO isn't currently supported in the GENERIC kernel.
- `ecn` — enable ECN (Explicit Congestion Notification) on this queue. ECN implies RED.

The CBQ scheduler supports an additional option:

- `borrow` — the queue can borrow bandwidth from the parent.

The HFSC scheduler supports some additional options (*sc* is an acronym for service curve):

- `realtime` *sc* — the minimum required bandwidth for the queue.
- `upperlimit` *sc* — the maximum allowed bandwidth for the queue.
- `linkshare` *sc* — the bandwidth share of a backlogged queue.

The format for service curve specifications is (*m1*, *d*, *m2*). The *m2* variable controls the bandwidth assigned to the queue, while *m1* and *d* are optional and can be used to control the initial bandwidth assignment. For the first *d* milliseconds, the queue gets the bandwidth given as *m1*, and afterward, the value given in *m2*.

Furthermore, with CBQ and HFSC, child queues can be specified as in an `altq` declaration, thus building a tree of queues using a part of their parent's bandwidth.

Packets can be assigned to queues based on filter rules by using the `queue` keyword. Normally only one queue is specified; when a second one is

Copyright © 2014, QNX Software Systems Limited

specified, it's used instead for packets that have a TOS of `lowdelay` and for TCP ACKs with no data payload.

To continue the previous example, the examples below specify the four referenced queues, plus a few child queues. The queues may then be referenced by filtering rules (see "*Packet filtering* (p. 1490)," below).

```
queue std bandwidth 10% cbq(default)
queue http bandwidth 60% priority 2 cbq(borrow red) \
      { employees, developers }
queue  developers bandwidth 75% cbq(borrow)
queue  employees bandwidth 15%
queue mail bandwidth 10% priority 0 cbq(borrow ecn)
queue ssh bandwidth 20% cbq(borrow) { ssh_interactive, ssh_bulk }
queue  ssh_interactive bandwidth 50% priority 7 cbq(borrow)
queue  ssh_bulk bandwidth 50% priority 0 cbq(borrow)

block return out on dc0 inet all queue std
pass out on dc0 inet proto tcp from $developerhosts to any port 80 \
      keep state queue developers
pass out on dc0 inet proto tcp from $employeehosts to any port 80 \
      keep state queue employees
pass out on dc0 inet proto tcp from any to any port 22 \
      keep state queue(ssh_bulk, ssh_interactive)
pass out on dc0 inet proto tcp from any to any port 25 \
      keep state queue mail
```

Translation

Translation rules modify either the source or destination address of the packets associated with a stateful connection. A stateful connection is automatically created to track packets matching such a rule as long as they aren't blocked by the filtering section of `pf.conf`. The translation engine modifies the specified address and/or port in the packet, recalculates IP, TCP and UDP checksums as necessary, and passes it to the packet filter for evaluation.

Since translation occurs before filtering the filter engine will see packets as they look after any addresses and ports have been translated. Filter rules will therefore have to filter based on the translated address and port number. Packets that match a translation rule are automatically passed only if the `pass` modifier is given; otherwise they're still subject to block and pass rules.

The state entry created permits `pf` to keep track of the original address for traffic associated with that state and correctly direct return traffic for that connection.

Various types of translation are possible with `pf`:

`binat`

Specifies a bidirectional mapping between an external IP netblock and an internal IP netblock.

`nat`

Specifies that IP addresses are to be changed as the packet traverses the given interface. This technique allows one or more IP addresses on the

Copyright © 2014, QNX Software Systems Limited

translating host to support network traffic for a larger range of machines on an "inside" network. Although in theory any IP address can be used on the inside, it is strongly recommended that one of the address ranges defined by RFC 1918 be used. These netblocks are:

- 10.0.0.0–10.255.255.255 (all of net 10, i.e. 10/8)
- 172.16.0.0–172.31.255.255 (i.e., 172.16/12)
- 192.168.0.0–192.168.255.255 (i.e., 192.168/16)

`rdr`

The packet is redirected to another destination and possibly a different port. The `rdr` rules can optionally specify port ranges instead of single ports. For example:

```
rdr ... port 2000:2999 -> ... port 4000
```

redirects ports 2000 to 2999 (inclusive) to port 4000. This command:

```
rdr ... port 2000:2999 -> ... port 4000:*
```

redirects port 2000 to 4000, 2001 to 4001, ..., 2999 to 4999.

In addition to modifying the address, some translation rules may modify source or destination ports for TCP or UDP connections; implicitly in the case of `nat` rules and explicitly in the case of `rdr` rules. Port numbers are never translated with a `binat` rule.

For each packet processed by the translator, the translation rules are evaluated in sequential order, from first to last. The first matching rule decides what action is taken.

The `no` option prefixed to a translation rule causes packets to remain untranslated, much in the same way as `drop quick` works in the packet filter (see below). If no rule matches the packet, the packet is passed to the filter engine unmodified.

Translation rules apply only to packets that pass through the specified interface, and if no interface is specified, translation is applied to packets on all interfaces. For instance, redirecting port 80 on an external interface to an internal web server works only for connections originating from the outside. Connections to the address of the external interface from local hosts aren't redirected, since such packets don't actually pass through the external interface. Redirections can't reflect packets back through the interface they arrive on; they can only be redirected to hosts connected to different interfaces or to the firewall itself.

Note that redirecting external incoming connections to the loopback address, as in:

```
rdr on ne3 inet proto tcp to port 8025 -> 127.0.0.1 port 25
```

Copyright © 2014, QNX Software Systems Limited

effectively allows an external host to connect to daemons bound solely to the loopback address, circumventing the traditional blocking of such connections on a real interface. Unless this effect is desired, any of the local non-loopback addresses should be used as redirection target instead, which allows external connections only to daemons bound to this address or not bound to any address.

See "*Translation examples* (p. 1507)," below.

Packet filtering

The `pf` pseudo-device can block and pass packets based on attributes of their layer-3 (see IP and IPv6 in the QNX Neutrino *C Library Reference*) and layer-4 (see ICMP, ICMP6, TCP, and UDP) headers. In addition, packets may also be assigned to queues for the purpose of bandwidth control.

For each packet processed by the packet filter, the filter rules are evaluated in sequential order, from first to last. The last matching rule decides what action is taken.

You can use the following actions in the filter:

`block`

Block the packet. There are a number of ways in which a block rule can behave when blocking a packet. The default behavior is to drop packets silently, however you can override this or make it explicit either globally, by setting the block-policy option, or on a per-rule basis with one of the following options:

- `drop` — silently drop the packet.
- `return-rst` — this applies only to TCP packets, and issues a TCP RST that closes the connection.
- `return-icmp`, `return-icmp6` — cause ICMP messages to be returned for packets that match the rule. By default, this is an ICMP UNREACHABLE message, however you can override this by specifying a message as a code or number.
- `return` — cause a TCP RST to be returned for TCP packets and an ICMP UNREACHABLE for UDP and other packets.

Options returning ICMP packets currently have no effect if `pf` operates on a *`bridge`*, as the code to support this feature hasn't yet been implemented.

`pass`

Pass the packet.

If no rule matches the packet, the default action is to pass it. To block everything by default and pass only packets that match explicit rules, use:

```
block all
```


Copyright © 2014, QNX Software Systems Limited

as the first filter rule.

See "*Filter examples* (p. 1509)," below.

Parameters

The rule parameters specify the packets to which a rule applies. A packet always comes in on, or goes out through, one interface. Most parameters are optional. If a parameter is specified, the rule applies only to packets with matching attributes. Certain parameters can be expressed as lists, in which case `pfctl` generates all needed rule combinations.

`in` or `out`

This rule applies to incoming or outgoing packets. If you specify neither `in` nor `out`, the rule matches packets in both directions.

`log`

In addition to the action specified, generate a log message. All packets for that connection are logged, unless the `keep state`, `modulate state` or `synproxy state` options are specified, in which case only the packet that establishes the state is logged. (See `keep state`, `modulate state` and `synproxy state` below). The logged packets are sent to the *`pflog`* interface. This interface is monitored by the *`pflogd`* logging daemon, which dumps the logged packets to the file `/var/log/pflog` in *`pcap`* binary format.

`log-all`

Used with `keep state`, `modulate state` or `synproxy state` rules to force logging of all packets for a connection. As with `log`, packets are logged to *`pflog`*.

`quick`

If a packet matches a rule that has the `quick` option set, this rule is considered the last matching rule, and evaluation of subsequent rules is skipped.

on *interface*

This rule applies only to packets coming in on, or going out through, this particular interface. It's also possible to simply give the interface driver name, such as `ppp` or `fxp`, to make the rule match packets flowing through a group of interfaces.

af

Copyright © 2014, QNX Software Systems Limited

This rule applies only to packets of this address family. Supported values are `inet` and `inet6`.

`proto` ***protocol***

This rule applies only to packets of this protocol. Common protocols are ICMP, ICMP6, TCP, and UDP. For a list of all the protocol name-to-number mappings used by `pfctl`, see the file `/etc/protocols`.

`from` ***source*** **`port`** ***source*** **`os`** ***source*** **`to`** ***dest*** **`port`** ***dest***

This rule applies only to packets with the specified source and destination addresses and ports. You can specify addresses in CIDR notation (matching netblocks), as symbolic host names or interface names, or as any of the following keywords:

- `any` — any address.
- `route` *label* — any address whose associated route has the label specified by *label*. See the ROUTE protocol (in the QNX Neutrino *C Library Reference*) and the *`route`* (p. 1680) utility.
- `no-route` — any address that isn't currently routable.
- *table* — any address that matches the given table.

Interface names can have modifiers appended:

- `:network` — translates to the network(s) attached to the interface.
- `:broadcast` — translates to the interface's broadcast address(es).
- `:peer` — translates to the point-to-point interface's peer address(es).
- `:0` — don't include interface aliases.

Host names may also have the `:0` option appended to restrict the name resolution to the first of each v4 and v6 address found.

Host-name resolution and interface-to-address translation are done when the rule set is loaded. When the address of an interface (or host name) changes (under DHCP or PPP, for instance), the rule set must be reloaded for the change to be reflected in `io-pkt`.

Surrounding the interface name (and optional modifiers) in parentheses changes this behavior: the rule is automatically updated whenever the interface changes its address. The rule set doesn't need to be reloaded. This is especially useful with `nat`.

You can specify ports either by number or by name. For example, port 80 can be specified as `www`. For a list of all port name-to-number mappings used by `pfctl`, see the file `/etc/services`.

You can use these operators to specify ports and ranges of ports:

Copyright © 2014, QNX Software Systems Limited

Operator	Meaning
=	Equal to
!=	Unequal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
:	Range, including boundaries
><	Range, excluding boundaries
<>	Except range

The ><, <> and : operators are binary; they take two arguments. For instance:

This:	Means:
`port 2000:2004`	All ports 2000 and 2004; hence ports 2000, 2001, 2002, 2003 and 2004
`port 2000 >< 2004`	All ports > 2000 and < 2004; hence ports 2001, 2002 and 2003
`port 2000 <> 2004`	All ports < 2000 or > 2004; hence ports 1-1999 and 2005-65535

The operating system of the source host can be specified in the case of TCP rules with the OS modifier. See "*Operating system fingerprinting* (p. 1503)" for more information.

The host, port and OS specifications are optional, as in the following examples:

```
pass in all
pass in from any to any
pass in proto tcp from any port <= 1024 to any
pass in proto tcp from any to any port 25
pass in proto tcp from 10.0.0.0/8 port > 1024 \
      to ! 10.1.2.3 port != ssh
pass in proto tcp from any os "OpenBSD" flags S/SA
pass in proto tcp from route "DTAG"
```

`all`

This is equivalent to `from any to any`.

`group` ***group***

Copyright © 2014, QNX Software Systems Limited

This functionality isn't supported in this version of NetBSD.

user *user*

This rule applies only to packets of sockets owned by the specified user. For outgoing connections initiated from the firewall, this is the user that opened the connection. For incoming connections to the firewall itself, this is the user that listens on the destination port. For forwarded connections, where the firewall isn't a connection endpoint, the user and group are unknown.

All packets, both outgoing and incoming, of one connection are associated with the same user and group. Only TCP and UDP packets can be associated with users; for other protocols these parameters are ignored.

User and group refer to the effective (as opposed to the real) IDs, in case the socket is created by a `setuid` or `setgid` process. User and group IDs are stored when a socket is created; when a process creates a listening socket as `root` (for instance, by binding to a privileged port) and subsequently changes to another user ID (to drop privileges), the credentials will remain `root`.

You can specify user and group IDs as either numbers or names. The syntax is similar to the one for ports. The value `unknown` matches packets of forwarded connections. You can use `unknown` only with the operators = and `!=`. Other constructions, such as `user >= unknown`, are invalid. Forwarded packets with unknown user and group ID match only rules that explicitly compare against `unknown` with the operators = or `!=`. For instance, `user >= 0` doesn't match forwarded packets. The following example allows only selected users to open outgoing connections:

```
block out proto { tcp, udp } all
pass  out proto { tcp, udp } all \
      user { < 1000, dhartmei } keep state
```

flags *a*/*b* | /*b*

This rule applies only to TCP packets that have the flags *a* set out of set *b*. Flags not specified in *b* are ignored. The flags are: (F)IN, (S)YN, (R)ST, (P)USH, (A)CK, (U)RG, (E)CE, and C(W)R. For example:

This:	Means:
`flags S/S`	Flag SYN is set. The other flags are ignored.
`flags S/SA`	Out of SYN and ACK, exactly SYN may be set. SYN, SYN+PSH and SYN+RST match, but SYN+ACK,

Copyright © 2014, QNX Software Systems Limited

This:	Means:
	ACK and ACK+RST don't. This is more restrictive than the previous example.
`flags /SFRA`	If the first set isn't specified, it defaults to none. All of SYN, FIN, RST and ACK must be unset.

`icmp-type` *type* `code` *code* or `icmp6-type` *type* `code` *code*

This rule applies only to ICMP or ICMPv6 packets with the specified type and code. Text names for ICMP types and codes are listed in the entries for ICMP and ICMP6 in the QNX Neutrino *C Library Reference*. This parameter is valid only for rules that cover protocols ICMP or ICMP6. The protocol and the ICMP type indicator (`icmp-type` or `icmp6-type`) must match.

`tos` *string* | *number*

This rule applies to packets with the specified TOS bits set. TOS may be given as one of `lowdelay`, `throughput`, `reliability`, or as either hexadecimal or decimal. For example, the following rules are identical:

- `pass all tos lowdelay`
- `pass all tos 0x10`
- `pass all tos 16`

`allow-opts`

By default, packets that contain IP options are blocked. When you specify `allow-opts` for a `pass` rule, packets that pass the filter based on that rule (last matching) do so even if they contain IP options. For packets that match a state, the rule that initially created the state is used. The implicit `pass` rule that's used when a packet doesn't match any rules doesn't allow IP options.

`label` *string*

Add a label (name) to the rule, which you can use to identify the rule. For instance, `pfctl -s labels` shows per-rule statistics for rules that have labels.

You can use the following macros in labels:

Copyright © 2014, QNX Software Systems Limited

Macro	Meaning
`$if`	The interface
`$srcaddr`	The source IP address
`$dstaddr`	The destination IP address
`$srcport`	The source port specification
`$dstport`	The destination port specification
`$proto`	The protocol name
`$nr`	The rule number

For example:

```
ips = "{ 1.2.3.4, 1.2.3.5 }"
pass in proto tcp from any to $ips \
     port > 1023 label "$dstaddr:$dstport"
```

expands to:

```
pass in inet proto tcp from any to 1.2.3.4 \
     port > 1023 label "1.2.3.4:>1023"
pass in inet proto tcp from any to 1.2.3.5 \
     port > 1023 label "1.2.3.5:>1023"
```

The macro expansion for the label directive occurs only when the configuration file is parsed, not during runtime.

queue ***queue*** **|** **(*queue*, *queue*)**

Packets matching this rule are assigned to the specified queue. If you specify two queues, packets that have a TOS of `lowdelay` and TCP ACKs with no data payload are assigned to the second one. See “*Queueing* (p. 1484),” for setup details. For example:

```
pass in proto tcp to port 25 queue mail
pass in proto tcp to port 22 queue(ssh_bulk, ssh_prio)
```

`tag` ***string***

Packets matching this rule are tagged with the specified string. The tag acts as an internal marker that can be used to identify these packets later on. You can use this, for example, to provide trust between interfaces and to determine if packets have been processed by translation rules.

Tags are “sticky”, meaning that the packet is tagged even if the rule isn't the last matching rule. Further matching rules can replace the tag with a new one, but don't remove a previously applied tag. A packet is only ever

Copyright © 2014, QNX Software Systems Limited

assigned one tag at a time. Any `pass` rules that use the `tag` keyword must also use `keep state`, `modulate state` or `synproxy state`.

You can tag packets during `nat`, `rdr`, or `binat` rules in addition to filter rules. Tags take the same macros as labels (see above).

`tagged` ***string***

Used with filter or translation rules to specify that packets must already be tagged with the given tag in order to match the rule. You can also do inverse tag matching by specifying the `!` operator before the `tagged` keyword.

`probability` ***number***

A probability attribute can be attached to a rule, with a value set between 0 and 1, bounds not included. In that case, the rule is honored using the given probability value only. For example, the following rule drops 20% of incoming ICMP packets:

```
block in proto icmp probability 20%
```

Routing

If a packet matches a rule with a route option set, the packet filter routes the packet according to the type of route option. When such a rule creates state, the route option is also applied to all packets matching the same connection.

`fastroute`

Do a normal route lookup to find the next hop for the packet.

`route-to`

Route the packet to the specified interface with an optional address for the next hop. When a `route-to` rule creates state, only packets that pass in the same direction as the filter rule specifies are routed in this way. Packets passing in the opposite direction (replies) aren't affected and are routed normally.

`reply-to`

Similar to `route-to`, but routes packets that pass in the opposite direction (replies) to the specified interface. Opposite direction is defined only in the context of a state entry, and `reply-to` is useful only in rules that create state. You can use it on systems with multiple external connections to route all outgoing packets of a connection through the interface the incoming connection arrived through (symmetric routing enforcement).

`dup-to`

Copyright © 2014, QNX Software Systems Limited

Create a duplicate of the packet and route it like `route-to`. The original packet gets routed as it normally would.

Pool options

For `nat` and `rdr` rules, (as well as for the `route-to`, `reply-to` and `dup-to` rule options) for which there is a single redirection address that has a subnet mask smaller than 32 for IPv4 or 128 for IPv6 (more than one IP address), you can use a variety of different methods for assigning this address:

`bitmask`

Apply the network portion of the redirection address to the address to be modified (source with `nat`, destination with `rdr`).

`random`

Select an address at random within the defined block of addresses.

`source-hash`

Use a hash of the source address to determine the redirection address, ensuring that the redirection address is always the same for a given source. You can optionally specify a key after this keyword either in hexadecimal or as a string; by default, `pfctl` randomly generates a key for `source-hash` every time the rule set is reloaded.

`round-robin`

Loop through the redirection address(es).

When more than one redirection address is specified, `round-robin` is the only permitted pool type.

`static-port`

With `nat` rules, prevent `pf` from modifying the source port on TCP and UDP packets.

Additionally, you can specify the `sticky-address` option to help ensure that multiple connections from the same source are mapped to the same redirection address. You can use this option with the `random` and `round-robin` pool options. Note that by default, these associations are destroyed as soon as there are no longer states that refer to them; in order to make the mappings last beyond the lifetime of the states, increase the global options with `set timeout source-track`. See "*Stateful tracking options* (p. 1501)" for more ways to control the source tracking.

Stateful inspection

The `pf` packet filter is *stateful*, which means it can track the state of a connection. Instead of passing all traffic to port 25, for instance, it's possible to pass only the

Copyright © 2014, QNX Software Systems Limited

initial packet, and then begin to keep state. Subsequent traffic will flow because the filter is aware of the connection.

If a packet matches a `pass ... keep` state rule, the filter creates a state for this connection and automatically lets pass all subsequent packets of that connection.

Before any rules are evaluated, the filter checks whether the packet matches any state. If it does, the packet is passed without evaluation of any rules.

States are removed after the connection is closed or has timed out.

This has several advantages:

- Comparing a packet to a state involves checking its sequence numbers. If the sequence numbers are outside the narrow windows of expected values, the packet is dropped. This prevents spoofing attacks, such as when an attacker sends packets with a fake source address/port but doesn't know the connection's sequence numbers.
- Looking up states is usually faster than evaluating rules. If there are 50 rules, all of them are evaluated sequentially in O(n). Even with 50000 states, only 16 comparisons are needed to match a state, since states are stored in a binary search tree that allows searches in O(log2 n).

For instance:

```
block all
pass out proto tcp from any to any flags S/SA keep state
pass in  proto tcp from any to any port 25 flags S/SA keep state
```

This rule set blocks everything by default. Only outgoing connections and incoming connections to port 25 are allowed. The initial packet of each connection has the SYN flag set, will be passed and creates state. All further packets of these connections are passed if they match a state.

By default, packets coming in and out of any interface can match a state, but it's also possible to change that behavior by assigning states to a single interface or a group of interfaces.

The default policy is specified by the state-policy global option, but you can adjust this on a per-rule basis by adding one of the `if-bound`, `group-bound`, or `floating` keywords to the `keep state` option. For example, if a rule is defined as:

```
pass out on ppp from any to 10.12/16 keep state (group-bound)
```

a state created on `ppp0` would match packets an all PPP interfaces, but not packets flowing through `fxp0` or any other interface.

Keeping rules floating is the more flexible option when the firewall is in a dynamic routing environment. However, this has some security implications, since a state created by one trusted network could allow potentially hostile packets coming in from other interfaces.

Copyright © 2014, QNX Software Systems Limited

Specifying flags S/SA restricts state creation to the initial SYN packet of the TCP handshake. You can also be less restrictive, and allow state creation from intermediate (non-SYN) packets. This causes `pf` to synchronize to existing connections, for instance if you flush the state table.

For UDP, which is stateless by nature, `keep state` creates state as well. UDP packets are matched to states using only host addresses and ports.

ICMP messages fall into two categories: ICMP error messages, which always refer to a TCP or UDP packet, are matched against the referred to connection. If you keep state on a TCP connection, and an ICMP source quench message referring to this TCP connection arrives, it will be matched to the right state and get passed.

For ICMP queries, `keep state` creates an ICMP state, and `pf` knows how to match ICMP replies to states. For example:

```
pass out inet proto icmp all icmp-type echoreq keep state
```

allows echo requests (such as those created by *ping* (p. 1543)) out, creates state, and matches incoming echo replies correctly to states.

The `nat`, `binat`, and `rdr` rules implicitly create state for connections.

State modulation

Much of the security derived from TCP is attributable to how well the initial sequence numbers (ISNs) are chosen. Some popular stack implementations choose very poor ISNs, and thus are normally susceptible to ISN prediction exploits. By applying a modulate state rule to a TCP connection, `pf` creates a high-quality random sequence number for each connection endpoint.

The `modulate state` directive implicitly keeps state on the rule and is applicable only to TCP connections. For instance:

```
block all
pass out proto tcp from any to any modulate state
pass in  proto tcp from any to any port 25 flags S/SA modulate state
```

There are some caveats associated with state modulation:

- You can't apply a `modulate state` rule to a pre-existing but unmodulated connection. Such an application would desynchronize TCP's strict sequencing between the two endpoints. Instead, `pf` treats the `modulate state` modifier as a `keep state` modifier, and the pre-existing connection is inferred without the protection conferred by modulation.
- The other caveat affects currently modulated states when the state table is lost (firewall reboot, flushing the state table, etc...). The `pf` packet filter can't infer a connection again after the state table flushes the connection's modulator. When the state is lost, the connection may be left dangling until the respective endpoints

Copyright © 2014, QNX Software Systems Limited

time out the connection. It's possible on a fast local network for the endpoints to start an ACK storm while trying to resynchronize after the loss of the modulator. Using a `flags S/SA` modifier on `modulate state` rules between fast networks is suggested to prevent ACK storms.

SYN proxy

By default, `pf` passes packets that are part of a TCP handshake between the endpoints. You can use the `synproxy state` option to cause `pf` itself to complete the handshake with the active endpoint, perform a handshake with the passive endpoint, and then forward packets between the endpoints.

No packets are sent to the passive endpoint before the active endpoint has completed the handshake, hence so-called SYN floods with spoofed source addresses will not reach the passive endpoint, as the sender can't complete the handshake.

The proxy is transparent to both endpoints, they each see a single connection from/to the other endpoint. The `pf` packet filter chooses random initial sequence numbers for both handshakes. Once the handshakes are completed, the sequence number modulators (see previous section) are used to translate further packets of the connection. Hence, `synproxy state` includes `modulate state` and `keep state`.

Rules with `synproxy` won't work if `pf` operates on a *`bridge`*.

Example:

```
pass in proto tcp from any to any port www flags S/SA synproxy state
```

Stateful tracking options

All of `keep state`, `modulate state` and `synproxy state` support the following options:

`max` *number*

Limit the number of concurrent states the rule may create. When this limit is reached, further packets matching the rule that would create state are dropped, until existing states time out.

timeout seconds

Change the timeout values used for states created by this rule. For a list of all valid timeout names, see "*Options* (p. 1478)", above.

You can specify multiple options, separated by commas:

```
pass in proto tcp from any to any \
       port www flags S/SA keep state \
       (max 100, source-track rule, max-src-nodes 75, \
       max-src-states 3, tcp.established 60, tcp.closing 5)
```

If you specify the `source-track` keyword, the number of states per source IP is tracked:

Copyright © 2014, QNX Software Systems Limited

source-track rule

The maximum number of states created by this rule is limited by the rule's `max-src-nodes` and `max-src-state` options. Only state entries created by this particular rule count toward the rule's limits.

source-track global

The number of states created by all rules that use this option is limited. Each rule can specify different `max-src-nodes` and `max-src-states` options, however state entries created by any participating rule count towards each individual rule's limits.

You can set the following limits:

max-src-nodes *number*

Limit the maximum number of source addresses that can simultaneously have state table entries.

max-src-states *number*

Limit the maximum number of simultaneous state entries that a single source address can create with this rule.

For stateful TCP connections, limits on established connections (those that have completed the TCP 3-way handshake) can also be enforced per source IP.

max-src-conn *number*

Limit the maximum number of simultaneous TCP connections that have completed the 3-way handshake that a single host can make.

max-src-conn-rate *number* / *seconds*

Limit the rate of new connections over a time interval. The connection rate is an approximation calculated as a moving average.

Because the 3-way handshake ensures that the source address isn't being spoofed, more aggressive action can be taken based on these limits. With the overload *table* state option, source IP addresses that hit either of the limits on established connections will be added to the named table. You can use this table in the rule set to block further activity from the offending host, redirect it to a tarpit process, or restrict its bandwidth.

The optional `flush` keyword kills all states created by the matching rule that originate from the host that exceeds these limits. The `global` modifier to the `flush` command kills all states originating from the offending host, regardless of which rule created the state.

Copyright © 2014, QNX Software Systems Limited

For example, the following rules protect the web server against hosts making more than 100 connections in 10 seconds. Any host that connects faster than this rate will have its address added to the *bad_hosts* table and have all states originating from it flushed. Any new packets arriving from this host will be dropped unconditionally by the `block` rule.

```
block quick from bad_hosts
pass in on $ext_if proto tcp to $webserver port www flags S/SA keep state \
      (max-src-conn-rate 100/10, overload bad_hosts flush global)
```

Operating system fingerprinting

Passive OS Fingerprinting is a mechanism to inspect nuances of a TCP con nection's initial SYN packet and guess at the host's operating system. Unfortunately these nuances are easily spoofed by an attacker so the fingerprint isn't useful in making security decisions. But the fingerprint is typically accurate enough to make policy decisions upon.

The fingerprints may be specified by operating system class, by version, or by subtype/patchlevel. The class of an operating system is typically the vendor or genre and would be OpenBSD for the `pf` firewall itself. The version of the oldest available OpenBSD release on the main ftp site would be 2.6 and the fingerprint would be written

```
"OpenBSD 2.6"
```

The subtype of an operating system is typically used to describe the patchlevel if that patch led to changes in the TCP stack behavior. In the case of OpenBSD, the only subtype is for a fingerprint that was normalized by the `no-df` scrub option and would be specified as:

```
"OpenBSD 3.3 no-df"
```

Fingerprints for most popular operating systems are provided by `/etc/pf.os`. Once `pf` is running, you can get a complete list of known operating system finger listed by running:

```
# pfctl -so
```

Filter rules can enforce policy at any level of operating system specification assuming a fingerprint is present. Policy could limit traffic to approved operating systems or even ban traffic from hosts that aren't at the latest service pack.

You can also use the `unknown` class as the fingerprint that matches packets for which no operating system fingerprint is known.

Examples:

```
pass  out proto tcp from any os OpenBSD keep state
block out proto tcp from any os Doors
block out proto tcp from any os "Doors PT"
block out proto tcp from any os "Doors PT SP3"
block out from any os "unknown"
pass on lo0 proto tcp from any os "OpenBSD 3.3 lo0" keep state
```

Copyright © 2014, QNX Software Systems Limited

Operating system fingerprinting is limited only to the TCP SYN packet. This means that it doesn't work on other protocols and doesn't match a currently established connection.

Operating system fingerprints are occasionally wrong. There are several problems:

- an attacker can trivially craft his packets to appear as any operating system he chooses
- an operating system patch could change the stack behavior, and no fingerprints will match it until the database is updated
- multiple operating systems may have the same fingerprint.

Blocking spoofed traffic

Spoofing is the faking of IP addresses, typically for malicious purposes. The `anti spoof` directive expands to a set of filter rules that block all traffic with a source IP from the network(s) directly connected to the specified interface(s) from entering the system through any other interface. For example, the line:

```
antispoof for lo0
```

expands to:

```
block drop in on ! lo0 inet from 127.0.0.1/8 to any
block drop in on ! lo0 inet6 from ::1 to any
```

For non-loopback interfaces, there are additional rules to block incoming packets with a source IP address identical to the interface's IP(s). For example, assuming the interface `wi0` has an IP address of 10.0.0.1 and a netmask of 255.255.255.0, the line:

```
antispoof for wi0 inet
```

expands to:

```
block drop in on ! wi0 inet from 10.0.0.0/24 to any
block drop in inet from 10.0.0.1 to any
```

Rules created by the `antispoof` directive interfere with packets sent over loopback interfaces to local addresses. You should pass these explicitly.

Fragment handling

The size of IP datagrams (packets) can be significantly larger than the maximum transmission unit (MTU) of the network. In cases when it is necessary or more efficient to send such large packets, the large packet will be fragmented into many smaller packets that will each fit onto the wire. Unfortunately for a firewalling device, only the first logical fragment will contain the necessary header information for the subprotocol that allows `pf` to filter on things such as TCP ports or to perform NAT.

Copyright © 2014, QNX Software Systems Limited

Besides the use of scrub rules as described in "*Traffic normalization* (p. 1483)" above, there are three options for handling fragments in the packet filter.

One alternative is to filter individual fragments with filter rules. If no scrub rule applies to a fragment, it's passed to the filter. Filter rules with matching IP header parameters decide whether the fragment is passed or blocked, in the same way as complete packets are filtered. Without reassembly, fragments can be filtered based only on IP header fields (source/destination address, protocol), since subprotocol header fields aren't available (TCP/UDP port numbers, ICMP code/type). You can use the `fragment` option to restrict filter rules to apply only to fragments, but not complete packets. Filter rules without the `fragment` option still apply to fragments, if they only specify IP header fields. For instance, the rule:

```
pass in proto tcp from any to any port 80
```

never applies to a fragment, even if the fragment is part of a TCP packet with destination port 80, because without reassembly this information isn't available for each fragment. This also means that fragments can't create new or match existing state table entries, which makes stateful filtering and address translation (NAT, redirection) for fragments impossible.

It's also possible to reassemble only certain fragments by specifying source or destination addresses or protocols as parameters in scrub rules.

In most cases, the benefits of reassembly outweigh the additional memory cost, and it's recommended you use `set reassemble yes` to reassemble all fragments.

You can limit the memory allocated for fragment caching by using `pfctl`. Once this limit is reached, fragments that would have to be cached are dropped until other entries time out. You can also adjust the timeout value.

Anchors

Besides the main rule set, `pfctl` can load rule sets into anchor attachment points. An anchor is a container that can hold rules, address tables, and other anchors.

An anchor has a name that specifies the path where you can use `pfctl` to access the anchor to perform operations on it, such as attaching child anchors to it or loading rules into it. Anchors may be nested, with components separated by slashes (`/`), similar to how file system hierarchies are laid out. The main rule set is actually the default anchor, so filter and translation rules, for example, may also be contained in any anchor.

An anchor can reference another anchor attachment point using the following kinds of rules:

`nat-anchor` *name*

Evaluate the `nat` rules in the specified anchor.

Copyright © 2014, QNX Software Systems Limited

rdr-anchor *name*

Evaluate the `rdr` rules in the specified anchor.

binat-anchor *name*

Evaluate the `binat` rules in the specified anchor.

anchor *name*

Evaluate the filter rules in the specified anchor.

load anchor *name* **from** *file*

Load the rules from the specified file into the anchor name.

When evaluation of the main rule set reaches an anchor rule, `pf` evaluates all rules specified in that anchor.

Matching filter and translation rules in anchors with the `quick` option are final and abort the evaluation of the rules in other anchors and the main rule set.

Anchor rules are evaluated relative to the anchor in which they are contained. For example, all anchor rules specified in the main rule set reference anchor attachment points underneath the main rule set, and anchor rules specified in a file loaded from a load anchor rule are attached under that anchor point.

Rules may be contained in anchor attachment points that don't contain any rules when the main rule set is loaded, and later such anchors can be manipulated through `pfctl` without reloading the main rule set or other anchors. For example:

```
ext_if = "kue0"
block on $ext_if all
anchor spam
pass out on $ext_if all keep state
pass in on $ext_if proto tcp from any \
to $ext_if port smtp keep state
```

blocks all packets on the external interface by default, then evaluates all rules in the anchor named `spam`, and finally passes all outgoing connections and incoming connections to port 25.

The following command loads a single rule into the anchor, which blocks all packets from a specific address:

```
# echo "block in quick from 1.2.3.4 to any" | \
        pfctl -a spam -f -
```

The anchor can also be populated by adding a `load anchor` rule after the anchor rule:

```
anchor spam
load anchor spam from "/etc/pf-spam.conf"
```

When `pfctl` loads `pf.conf`, it also loads all the rules from the file `/etc/pf-spam.conf` into the anchor.

Copyright © 2014, QNX Software Systems Limited

Optionally, anchor rules can specify the parameter's direction, interface, address family, protocol and source/destination address/port using the same syntax as filter rules. When parameters are used, the anchor rule is evaluated only for matching packets. This allows conditional evaluation of anchors, like:

```
block on $ext_if all
anchor spam proto tcp from any to any port smtp
pass out on $ext_if all keep state
pass in on $ext_if proto tcp from any to $ext_if port smtp keep state
```

The rules inside the anchor `spam` are evaluated only for TCP packets with destination port 25. Hence:

```
# echo "block in quick from 1.2.3.4 to any" | \
       pfctl -a spam -f -
```

blocks connections only from 1.2.3.4 to port 25.

Anchors may end with the asterisk (`*`) character, which signifies that all anchors attached at that point should be evaluated in the alphabetical ordering of their anchor name. For example:

```
anchor "spam/*"
```

evaluates each rule in each anchor attached to the `spam` anchor. Note that it evaluates only anchors that are directly attached to the `spam` anchor, and doesn't descend to evaluate anchors recursively.

Since anchors are evaluated relative to the anchor in which they are contained, there's a mechanism for accessing the parent and ancestor anchors of a given anchor. Similar to file system path name resolution, if the sequence `..` appears as an anchor path component, the parent anchor of the current anchor in the path evaluation at that point becomes the new current anchor. As an example, consider the following:

```
# echo ' anchor "spam/allowed" ' | pfctl -f -
# echo -e ' anchor "../banned" \n pass' | \
        pfctl -a spam/allowed -f -
```

Evaluation of the main rule set leads into the `spam/allowed` anchor, which evaluates the rules in the `spam/banned` anchor, if any, before finally evaluating the `pass` rule.

Since the parser specification for anchor names is a string, any reference to an anchor name containing slash (`/`) characters requires double quote (`"`) characters around the anchor name.

Translation examples

This example maps incoming requests on port 80 to port 8080, on which a daemon is running (because, for example, it isn't run as `root`, and therefore lacks permission to bind to port 80):

```
# use a macro for the interface name, so it can be changed easily
ext_if = "ne3"

# map daemon on 8080 to appear to be on 80
rdr on $ext_if proto tcp from any to any port 80 -> 127.0.0.1 port 8080
```

Copyright © 2014, QNX Software Systems Limited

If the pass modifier is given, packets matching the translation rule are passed without inspecting the filter rules:

```
rdr pass on $ext_if proto tcp from any to any port 80 -> 127.0.0.1 \
      port 8080
```

In the example below, vlan12 is configured as 192.168.168.1; the machine translates all packets coming from 192.168.168.0/24 to 204.92.77.111 when they're going out any interface except vlan12. This has the net effect of making traffic from the 192.168.168.0/24 network appear as though it is the Internet routable address 204.92.77.111 to nodes behind any interface on the router except for the nodes on vlan12. (Thus, 192.168.168.1 can talk to the 192.168.168.0/24 nodes.)

```
nat on ! vlan12 from 192.168.168.0/24 to any -> 204.92.77.111
```

In the example below, the machine sits between a fake internal 144.19.74.* network, and a routable external IP of 204.92.77.100. The no nat rule excludes protocol AH from being translated:

```
# NO NAT
no nat on $ext_if proto ah from 144.19.74.0/24 to any
nat on $ext_if from 144.19.74.0/24 to any -> 204.92.77.100
```

In the example below, packets bound for one specific server, as well as those generated by the system administrators aren't proxied; all other connections are:

```
# NO RDR
no rdr on $int_if proto { tcp, udp } from any to $server port 80
no rdr on $int_if proto { tcp, udp } from $sysadmins to any port 80
rdr on $int_if proto { tcp, udp } from any to any port 80 -> 127.0.0.1 \
      port 80
```

This longer example uses both a NAT and a redirection. The external interface has the address 157.161.48.183. On the internal interface, we are running *ftp-proxy*, listening for outbound ftp sessions captured to port 8021:

```
# NAT
# Translate outgoing packets' source addresses (any protocol).
# In this case, any address but the gateway's external address is mapped.
nat on $ext_if inet from ! ($ext_if) to any -> ($ext_if)

# NAT PROXYING
# Map outgoing packets' source port to an assigned proxy port instead of
# an arbitrary port.
# In this case, proxy outgoing isakmp with port 500 on the gateway.
nat on $ext_if inet proto udp from any port = isakmp to any -> ($ext_if) \
      port 500

# BINAT
# Translate outgoing packets' source address (any protocol).
# Translate incoming packets' destination address to an internal machine
# (bidirectional).
binat on $ext_if from 10.1.2.150 to any -> $ext_if

# RDR
# Translate incoming packets' destination addresses.
# As an example, redirect a TCP and UDP port to an internal machine.
rdr on $ext_if inet proto tcp from any to ($ext_if) port 8080 \
      -> 10.1.2.151 port 22
rdr on $ext_if inet proto udp from any to ($ext_if) port 8080 \
      -> 10.1.2.151 port 53

# RDR
# Translate outgoing ftp control connections to send them to localhost
```


Copyright © 2014, QNX Software Systems Limited

```
# for proxying with ftp-proxy(8) running on port 8021.
rdr on $int_if proto tcp from any to any port 21 -> 127.0.0.1 port 8021
```

In this example, a NAT gateway is set up to translate internal addresses using a pool of public addresses (192.0.2.16/28) and to redirect incoming web server connections to a group of web servers on the internal network:

```
# NAT LOAD BALANCE
# Translate outgoing packets' source addresses using an address pool.
# A given source address is always translated to the same pool address by
# using the source-hash keyword.
nat on $ext_if inet from any to any -> 192.0.2.16/28 source-hash

# RDR ROUND ROBIN
# Translate incoming web server connections to a group of web servers on
# the internal network.
rdr on $ext_if proto tcp from any to any port 80 \
      -> { 10.1.2.155, 10.1.2.160, 10.1.2.161 } round-robin
```

Filter examples

```
# The external interface is kue0
# (157.161.48.183, the only routable address)
# and the private network is 10.0.0.0/8, for which we are doing NAT.

# use a macro for the interface name, so it can be changed easily
ext_if = "kue0"

# normalize all incoming traffic
set reassemble on
scrub in on $ext_if all

# block and log everything by default
block return log on $ext_if all

# block anything coming from source we have no back routes for
block in from no-route to any

# block and log outgoing packets that don't have our address as source,
# they are either spoofed or something is misconfigured (NAT disabled,
# for instance), we want to be nice and don't send out garbage.
block out log quick on $ext_if from ! 157.161.48.183 to any

# silently drop broadcasts (cable modem noise)
block in quick on $ext_if from any to 255.255.255.255

# block and log incoming packets from reserved address space and invalid
# addresses, they are either spoofed or misconfigured, we can't reply to
# them anyway (hence, no return-rst).
block in log quick on $ext_if from { 10.0.0.0/8, 172.16.0.0/12, \
      192.168.0.0/16, 255.255.255.255/32 } to any

# ICMP

# pass out/in certain ICMP queries and keep state (ping)
# state matching is done on host addresses and ICMP ID (not type/code),
# so replies (like 0/0 for 8/0) will match queries
# ICMP error messages (which always refer to a TCP/UDP packet) are
# handled by the TCP/UDP states
pass on $ext_if inet proto icmp all icmp-type 8 code 0 keep state

# UDP

# pass out all UDP connections and keep state
pass out on $ext_if proto udp all keep state

# pass in certain UDP connections and keep state (DNS)
pass in on $ext_if proto udp from any to any port domain keep state

# TCP

# pass out all TCP connections and modulate state
pass out on $ext_if proto tcp all modulate state

# pass in certain TCP connections and keep state (SSH, SMTP, DNS, IDENT)
pass in on $ext_if proto tcp from any to any port { ssh, smtp, domain, \
```

Copyright © 2014, QNX Software Systems Limited

```
      auth } flags S/SA keep state

# pass in data mode connections for ftp-proxy running on this host.
# (see ftp-proxy(8) for details)
pass in on $ext_if proto tcp from any to 157.161.48.183 port >= 49152 \
      flags S/SA keep state

# Don't allow Windows 9x SMTP connections since they are typically
# a viral worm. Alternately we could limit these OSes to 1 connection each.
block in on $ext_if proto tcp from any os {"Windows 95", "Windows 98"} \
      to any port smtp

# Packet Tagging

# three interfaces: $int_if, $ext_if, and $wifi_if (wireless). NAT is
# being done on $ext_if for all outgoing packets. tag packets in on
# $int_if and pass those tagged packets out on $ext_if.  all other
# outgoing packets (i.e., packets from the wireless network) are only
# permitted to access port 80.

pass in on $int_if from any to any tag INTNET keep state
pass in on $wifi_if from any to any keep state

block out on $ext_if from any to any
pass out quick on $ext_if tagged INTNET keep state
pass out on $ext_if proto tcp from any to any port 80 keep state
```

Grammar

The syntax for `pf.conf` in BNF is as follows:

```
line           = ( option | pf-rule | nat-rule | binat-rule | rdr-rule |
                   antispoof-rule | altq-rule | queue-rule | anchor-rule |
                   trans-anchors | load-anchors | table-rule )

option         = "set" ( [ "timeout" ( timeout | "{" timeout-list "}" ) ] |
                   [ "optimization" [ "default" | "normal" |
                   "high-latency" | "satellite" |
                   "aggressive" | "conservative" ] ]
                   [ "limit" ( limit-item | "{" limit-list "}" ) ] |
                   [ "loginterface" ( interface-name | "none" ) ] |
                   [ "block-policy" ( "drop" | "return" ) ] |
                   [ "state-policy" ( "if-bound" | "group-bound" |
                   "floating" ) ]
                   [ "require-order" ( "yes" | "no" ) ]
                   [ "fingerprints" filename ] |
                   [ "debug" ( "none" | "urgent" | "misc" | "loud" ) ] |
                   [ "reassemble" ( "yes" | "no" ) ] )

pf-rule        = action [ ( "in" | "out" ) ]
                   [ "log" | "log-all" ] [ "quick" ]
                   [ "on" ifspec ] [ route ] [ af ] [ protospec ]
                   hosts [ filteropt-list ]

filteropt-list = filteropt-list filteropt | filteropt
filteropt      = user | flags | icmp-type | icmp6-type | tos |
                   ( "keep" | "modulate" | "synproxy" ) "state"
                   [ "(" state-opts ")" ] |
                   "fragment" | "no-df" | "min-ttl" number |
                   "max-mss" number | "random-id" | "reassemble tcp" |
                   "allow-opts" |
                   "label" string | "tag" string | [ ! ] "tagged" string
                   "queue" ( string | "(" string [ [ "," ] string ] ")" ) |
                   "probability" number"%"

nat-rule       = [ "no" ] "nat" [ "pass" ] [ "on" ifspec ] [ af ]
                   [ protospec ] hosts [ "tag" string ] [ "tagged" string ]
                   [ "->" ( redirhost | "{" redirhost-list "}" )
                   [ portspec ] [ pooltype ] [ "static-port" ] ]

binat-rule     = [ "no" ] "binat" [ "pass" ] [ "on" interface-name ]
                   [ af ] [ "proto" ( proto-name | proto-number ) ]
                   "from" address [ "/" mask-bits ] "to" ipspec
                   [ "tag" string ] [ "tagged" string ]
                   [ "->" address [ "/" mask-bits ] ]

rdr-rule       = [ "no" ] "rdr" [ "pass" ] [ "on" ifspec ] [ af ]
                   [ protospec ] hosts [ "tag" string ] [ "tagged" string ]
                   [ "->" ( redirhost | "{" redirhost-list "}" )
                   [ portspec ] [ pooltype ] ]

antispoof-rule = "antispoof" [ "log" ] [ "quick" ]
                   "for" ( interface-name | "{" interface-list "}" )
                   [ af ] [ "label" string ]

table-rule     = "table" "<" string ">" [ tableopts-list ]
tableopts-list = tableopts-list tableopts | tableopts
tableopts      = "persist" | "const" | "file" string |
                   "{" [ tableaddr-list ] "}"
tableaddr-list = tableaddr-list [ "," ] tableaddr-spec | tableaddr-spec
tableaddr-spec = [ "!" ] tableaddr [ "/" mask-bits ]
tableaddr      = hostname | ipv4-dotted-quad | ipv6-coloned-hex |
                   interface-name | "self"
```


Copyright © 2014, QNX Software Systems Limited

```
altq-rule      = "altq on" interface-name queueopts-list
                 "queue" subqueue
queue-rule     = "queue" string [ "on" interface-name ] queueopts-list
                 subqueue

anchor-rule    = "anchor" string [ ( "in" | "out" ) ] [ "on" ifspec ]
                 [ af ] [ "proto" ] [ protospec ] [ hosts ]

trans-anchors  = ( "nat-anchor" | "rdr-anchor" | "binat-anchor" ) string
                 [ "on" ifspec ] [ af ] [ "proto" ] [ protospec ] [ hosts ]

load-anchor    = "load anchor" string "from" filename

queueopts-list = queueopts-list queueopts | queueopts
queueopts      = [ "bandwidth" bandwidth-spec ] |
                 [ "qlimit" number ] | [ "tbrsize" number ] |
                 [ "priority" number ] | [ schedulers ]
schedulers     = ( cbq-def | priq-def | hfsc-def )
bandwidth-spec = "number" ( "b" | "Kb" | "Mb" | "Gb" | "%" )

action         = "pass" | "block" [ return ] | [ "no" ] "scrub"
return         = "drop" | "return" | "return-rst" [ "( ttl" number ")" ] |
                 "return-icmp" [ "(" icmpcode [ [ "," ] icmp6code ] ")" ] |
                 "return-icmp6" [ "(" icmp6code ")" ]
icmpcode       = ( icmp-code-name | icmp-code-number )
icmp6code      = ( icmp6-code-name | icmp6-code-number )

ifspec         = ( [ "!" ] interface-name ) | "{" interface-list "}"
interface-list = [ "!" ] interface-name [ [ "," ] interface-list ]
route          = "fastroute" |
                 ( "route-to" | "reply-to" | "dup-to" )
                 ( routehost | "{" routehost-list "}" )
                 [ pooltype ]
af             = "inet" | "inet6"

protospec      = "proto" ( proto-name | proto-number |
                 "{" proto-list "}" )
proto-list     = ( proto-name | proto-number ) [ [ "," ] proto-list ]

hosts          = "all" |
                 "from" ( "any" | "no-route" | "self" | host |
                 "{" host-list "}" | "route" string ) [ port ] [ os ]
                 "to"   ( "any" | "no-route" | "self" | host |
                 "{" host-list "}" | "route" string ) [ port ]

ipspec         = "any" | host | "{" host-list "}"
host           = [ "!" ] ( address [ "/" mask-bits ] | "<" string ">" )
redirhost      = address [ "/" mask-bits ]
routehost      = ( interface-name [ address [ "/" mask-bits ] ] )
address        = ( interface-name | "(" interface-name ")" | hostname |
                 ipv4-dotted-quad | ipv6-coloned-hex )
host-list      = host [ [ "," ] host-list ]
redirhost-list = redirhost [ [ "," ] redirhost-list ]
routehost-list = routehost [ [ "," ] routehost-list ]

port           = "port" ( unary-op | binary-op | "{" op-list "}" )
portspec       = "port" ( number | name ) [ ":" ( "*" | number | name ) ]
os             = "os"  ( os-name | "{" os-list "}" )
user           = "user" ( unary-op | binary-op | "{" op-list "}" )

unary-op       = [ "=" | "!=" | "<" | "<=" | ">" | ">=" ]
                 ( name | number )
binary-op      = number ( "<>" | "><" | ":" ) number
op-list        = ( unary-op | binary-op ) [ [ "," ] op-list ]

os-name        = operating-system-name
os-list        = os-name [ [ "," ] os-list ]

flags          = "flags" [ flag-set ] "/" flag-set
flag-set       = [ "F" ] [ "S" ] [ "R" ] [ "P" ] [ "A" ] [ "U" ] [ "E" ]
                 [ "W" ]

icmp-type      = "icmp-type" ( icmp-type-code | "{" icmp-list "}" )
icmp6-type     = "icmp6-type" ( icmp-type-code | "{" icmp-list "}" )
icmp-type-code = ( icmp-type-name | icmp-type-number )
                 [ "code" ( icmp-code-name | icmp-code-number ) ]
icmp-list      = icmp-type-code [ [ "," ] icmp-list ]

tos            = "tos" ( "lowdelay" | "throughput" | "reliability" |
                 [ "0x" ] number )

state-opts     = state-opt [ [ "," ] state-opts ]
state-opt      = ( "max" number | timeout |
                 "source-track" [ ( "rule" | "global" ) ] |
                 "max-src-nodes" number | "max-src-states" number |
                 "max-src-conn" number |
                 "max-src-conn-rate" number "/" number |
                 "overload" "<" string ">" [ "flush" ] |
                 "if-bound" | "group-bound" | "floating" )

timeout-list   = timeout [ [ "," ] timeout-list ]
timeout        = ( "tcp.first" | "tcp.opening" | "tcp.established" |
                 "tcp.closing" | "tcp.finwait" | "tcp.closed" |
                 "udp.first" | "udp.single" | "udp.multiple" |
                 "icmp.first" | "icmp.error" |
                 "other.first" | "other.single" | "other.multiple" |
                 "frag" | "interval" | "src.track" |
                 "adaptive.start" | "adaptive.end" ) number

limit-list     = limit-item [ [ "," ] limit-list ]
limit-item     = ( "states" | "frags" | "src-nodes" ) number

pooltype       = ( "bitmask" | "random" |
                 "source-hash" [ ( hex-key | string-key ) ] |
                 "round-robin" ) [ sticky-address ]
```

Copyright © 2014, QNX Software Systems Limited

```
subqueue       = string | "{" queue-list "}"
queue-list     = string [ [ "," ] string ]
cbq-def        = "cbq" [ "(" cbq-opt [ [ "," ] cbq-opt ] ")" ]
priq-def       = "priq" [ "(" priq-opt [ [ "," ] priq-opt ] ")" ]
hfsc-def       = "hfsc" [ "(" hfsc-opt [ [ "," ] hfsc-opt ] ")" ]
cbq-opt        = ( "default" | "borrow" | "red" | "ecn" | "rio" )
priq-opt       = ( "default" | "red" | "ecn" | "rio" )
hfsc-opt       = ( "default" | "red" | "ecn" | "rio" |
                   linkshare-sc | realtime-sc | upperlimit-sc )
linkshare-sc   = "linkshare" sc-spec
realtime-sc    = "realtime" sc-spec
upperlimit-sc  = "upperlimit" sc-spec
sc-spec        = ( bandwidth-spec |
                   "(" bandwidth-spec number bandwidth-spec ")" )
```

Associated files

/etc/hosts **(p. 946)**

Host name database.

/etc/pf.os

Default location of OS fingerprints.

/etc/protocols **(p. 1595)**

Protocol name database.

/etc/services **(p. 1744)**

Service name database.

/usr/share/examples/pf

Examples of rule sets.

Copyright © 2014, QNX Software Systems Limited

pfctl

Control the packet filter (PF) and network address translation (NAT) device

Syntax:

```
pfctl [-AdeghmNnOoqRrvz] [-a anchor] [-D macro=value]
      [-F modifier ] [-f file] [-i interface] [-k host]
      [-p device] [-s modifier]
      [-t table -T command [address ...]] [-x level]
```

Runs on:

QNX Neutrino

Options:

-A

Load only the queue rules present in the rule file. Other rules and options are ignored.

-a *anchor*

Apply the -f, -F, and -s options only to the rules in the specified anchor. In addition to the main rule set, `pfctl` can load and manipulate additional rule sets by name, called anchors. The main rule set is the default anchor.

Anchors are referenced by name and may be nested, with the various components of the anchor path separated by slashes (`/`), similar to how file system hierarchies are laid out. The last component of the anchor path is where rule-set operations are performed.

Evaluation of anchor rules from the main rule set is described in the documentation for `pf.conf`.

Private tables can also be put inside anchors, either by having table statements in the `pf.conf` file that is loaded in the anchor, or by using regular table commands, as in:

```
pfctl -a foo/bar -t mytable -T add 1.2.3.4 5.6.7.8
```

When a rule referring to a table is loaded in an anchor, the rule will use the private table if one is defined, and then fall back to the table defined in the main rule set, if there is one. This is similar to C rules for variable scope. It is possible to create distinct tables with the same name in the global rule set and in an anchor, but this is often bad design, and a warning is issued in that case.

Copyright © 2014, QNX Software Systems Limited

-D *macro=value*

Define *macro* to be set to *value* on the command line. This overrides the definition of *macro* in the rule set.

-d

Disable the packet filter.

-e

Enable the packet filter.

-F *modifier*

Flush the filter parameters specified by *modifier* (which you can abbreviate):

- `nat` — flush the NAT rules.
- `queue` — flush the queue rules.
- `rules` — flush the filter rules.
- `state` — flush the state table (NAT and filter).
- `Sources` — flush the source tracking table.
- `info` — flush the filter information (statistics that aren't bound to rules).
- `Tables` — flush the tables.
- `osfp` — flush the passive operating system fingerprints.
- `all` — flush all of the above.

-f *file*

Load the rules contained in the specified file. This file may contain macros, tables, options, and normalization, queueing, translation, and filtering rules. With the exception of macros and tables, the statements must appear in that order.

Specify `-` for the *file* to use standard input.

-g

Include output helpful for debugging.

-h

Display some help information.

-i *interface*

Restrict the operation to the given interface.

-k *host*

Copyright © 2014, QNX Software Systems Limited

Kill all of the state entries originating from the specified host. You can specify a second -k option, which will kill all the state entries from the first host to the second host. For example, to kill all of the state entries originating from `host`:

```
pfctl -k host
```

To kill all of the state entries from `host1` to `host2`:

```
pfctl -k host1 -k host2
```

-m

Merge in explicitly given options without resetting those that are omitted. This option lets you modify single options without disturbing the others:

```
echo "set loginterface fxp0" | pfctl -mf -
```

-N

Load only the NAT rules present in the rule file. Other rules and options are ignored.

-n

Don't actually load rules; just parse them.

-O

(“Oh”) Load only the options present in the rule file. Other rules and options are ignored.

-o

Enable the rule-set optimizer, which attempts to improve rule sets by removing rule duplication and making better use of rule ordering. Specifically, it does the following:

- removes duplicate rules
- removes rules that are a subset of another rule
- combines multiple rules into a table when advantageous
- reorders the rules to improve evaluation performance

You can specify a second -o option to use the currently loaded rule set as a feedback profile to tailor the optimization of the quick rules to the actual network behavior.

The rule-set optimizer modifies the rule set to improve performance. A side effect of the ruleset modification is that per-rule accounting

Copyright © 2014, QNX Software Systems Limited

statistics will have different meanings than before. If per-rule accounting is important (e.g. for billing purposes), either you shouldn't use the rule-set optimizer, or you should add a label field to all of the accounting rules to act as optimization barriers.

-p *device*

Use the device file *device* instead of the default, `/dev/pf`.

-q

Print only errors and warnings.

-R

Load only the filter rules present in the rule file. Other rules and options are ignored.

-r

Perform reverse DNS lookups on states when displaying them.

-s *modifier*

Show the filter parameters specified by *modifier* (which you can abbreviate):

- `nat` — show the currently loaded NAT rules.
- `queue` — show the currently loaded queue rules. When used together with -v, `pfctl` also shows per-queue statistics. When used together with -v -v, `pfctl` loops and shows updated queue statistics every five seconds, including measured bandwidth and packets per second.
- `rules` — show the currently loaded filter rules. When used together with -v, `pfctl` also shows the per-rule statistics (number of evaluations, packets and bytes).

  Note that the `skip step` optimization done automatically by `io-pkt` skips the evaluation of rules where possible. Packets passed statefully are counted in the rule that created the state (even though the rule isn't evaluated more than once for the entire connection).
- `Anchors` — show the currently loaded anchors directly attached to the main ruleset. If you also specify -a *anchor*, the anchors loaded directly below the given anchor are shown instead. If you specify -v, all anchors attached under the target anchor are displayed recursively.
- `state` — show the contents of the state table.
- `Sources` — show the contents of the source-tracking table.
- `info` — show filter information (statistics and counters). When used together with -v, source tracking statistics are also shown.

Copyright © 2014, QNX Software Systems Limited

- `labels` — show per-rule statistics (label, evaluations, packets, bytes) of filter rules with labels. This can be useful for accounting.
- `timeouts` — show the current global timeouts.
- `memory` — show the current pool memory hard limits.
- `Tables` — show the list of tables.
- `osfp` — show the list of operating system fingerprints.
- `Interfaces` — show the list of interfaces and interface drivers available to PF. When used together with a double -v option, `pfctl` also shows interface statistics. You can use the -i option to select an interface or a group of interfaces.
- `all` — show all of the above, except for the lists of interfaces and operating system fingerprints.

-T *command* [*address* ...]

Specify the command (which you can abbreviate) to apply to the table. The commands include:

- `kill` — kill a table.
- `flush` — flush all addresses of a table.
- `add` — add one or more addresses in a table. Automatically create a nonexisting table.
- `delete` — delete one or more addresses from a table.
- `replace` — replace the addresses of the table. Automatically create a nonexisting table.
- `show` — show the content (addresses) of a table.
- `test` — test if the given addresses match a table.
- `zero` — clear all the statistics of a table.
- `load` — load only the table definitions from `pf.conf`. You use this in conjunction with the -f option, as in:

```
pfctl -Tl -f pf.conf
```

For the `add`, `delete`, `replace`, and `test` commands, you can specify the list of addresses either directly on the command line and/or in an unformatted text file, using the -f flag. Comments starting with a # are allowed in the text file. With these commands, you can also use the -v option once or twice, in which case `pfctl` prints the detailed result of the operation for each individual address, prefixed by one of the following letters:

- A — the address/network has been added.
- C — the address/network has been changed (negated).

Copyright © 2014, QNX Software Systems Limited

- D — the address/network has been deleted.
- M — the address matches (test operation only).
- X — the address/network is duplicated and therefore ignored.
- Y — the address/network can't be added/deleted due to conflicting ! attributes.
- Z — the address/network has been cleared (statistics).

Each table maintains a set of counters that you can retrieve using the -v option. For example, the following commands define a wide-open firewall that keeps track of packets going to or coming from the OpenBSD FTP server. The following commands configure the firewall and send 10 pings to the FTP server:

```
printf "table <test> { ftp.NetBSD.org }\n \
      pass out to <test> keep state\n" | pfctl -f-
ping -qc10 ftp.NetBSD.org
```

We can now use the table `show` command to output, for each address and packet direction, the number of packets and bytes that are being passed or blocked by rules referencing the table. The time at which the current accounting started is also shown with the `Cleared` line:

```
pfctl -t test -vTshow
     129.128.5.191
        Cleared:     Thu Feb 13 18:55:18 2003
        In/Block:    [ Packets: 0        Bytes: 0        ]
        In/Pass:     [ Packets: 10       Bytes: 840      ]
        Out/Block:   [ Packets: 0        Bytes: 0        ]
        Out/Pass:    [ Packets: 10       Bytes: 840      ]
```

Similarly, you can view global information about the tables by using the -v option twice with the -s Tables command. This displays the number of addresses on each table, the number of rules that reference the table, and the global packet statistics for the whole table:

```
pfctl -vvsTables
--a-r-  test
    Addresses:   1
    Cleared:     Thu Feb 13 18:55:18 2003
    References:  [ Anchors: 0        Rules: 1        ]
    Evaluations: [ NoMatch: 3496     Match: 1        ]
    In/Block:    [ Packets: 0        Bytes: 0        ]
    In/Pass:     [ Packets: 10       Bytes: 840      ]
    In/XPass:    [ Packets: 0        Bytes: 0        ]
    Out/Block:   [ Packets: 0        Bytes: 0        ]
    Out/Pass:    [ Packets: 10       Bytes: 840      ]
    Out/XPass:   [ Packets: 0        Bytes: 0        ]
```

In this case, only one packet — the initial `ping` request — matched the table, but all packets passing as the result of the state are correctly accounted for. Reloading the table(s) or ruleset doesn't affect packet accounting in any way. The two `XPass` counters are incremented instead of the `Pass` counters when a “stateful” packet is passed but doesn't match the table anymore. This will happen in our example if someone were to flush the table while the `ping` command was running.

When used with a single -v, `pfctl` displays only the first line containing the table flags and name. The flags are defined as follows:

Copyright © 2014, QNX Software Systems Limited

- `c` — constant tables, which can't be altered outside `pf.conf`.
- `p` — persistent tables, which don't get automatically killed when no rules refer to them.
- `a` — tables that are part of the active table set. Tables without this flag don't really exist, can't contain addresses, and are listed only if you specify the -g option.
- `i` — tables that are part of the inactive table set. This flag can be witnessed only briefly during the loading of `pf.conf`.
- `r` — tables that are referenced (used) by rules.
- `h` — a table in the main rule set is hidden by one or more tables of the same name from anchors attached below it.

-t *table*

Specify the name of the table.

-v

Produce more verbose output. If you specify a second -v, `pfctl` produces even more verbose output including rule set warnings. See above for its effect on table commands.

-x *level*

Set the debugging level to one of the following (which you can abbreviate):

- `none` — don't generate debug messages.
- `urgent` — generate debug messages only for serious errors.
- `misc` — generate debug messages for various errors.
- `loud` — generate debug messages for common conditions.

-z

Clear per-rule statistics.

Description:

The `pfctl` utility communicates with the packet filter device using the *ioctl()* or *ioctl_socket()* interface described in *pf* (p. 1461). It lets you configure rule sets and parameters and retrieve status information from the packet filter.

Packet filtering restricts the types of packets that pass through network interfaces entering or leaving the host based on filter rules as described in *pf.conf* (p. 1476). The packet filter can also replace addresses and ports of packets. Replacing source addresses and ports of outgoing packets is called NAT (Network Address Translation) and is used to connect an internal network (usually reserved address space) to an

Copyright © 2014, QNX Software Systems Limited

external one (the Internet) by making all connections to external hosts appear to come from the gateway. Replacing destination addresses and ports of incoming packets is used to redirect connections to different hosts and/or ports. A combination of both translations, bidirectional NAT, is also supported. Translation rules are described in the documentation for `pf.conf`.

The packet filter doesn't itself forward packets between interfaces. Forwarding can be enabled by setting the *`sysctl`* (p. 1875) variables `net.inet.ip.forwarding` and/or `net.inet6.ip6.forwarding` to 1. Set them permanently in a file such as `/etc/sysctl.conf`, and then start `sysctl` using that file. For example, you could add this command:

```
sysctl -f /etc/sysctl.conf
```

to your system's `/etc/rc.d/rc.local` file.

Files:

`/etc/pf.conf` **(p. 1476)**

Packet filter rules file.

`/etc/pf.os`

Passive operating system fingerprint database.

Copyright © 2014, QNX Software Systems Limited

pidin

Display information about the processes in the system (QNX Neutrino)

Syntax:

```
pidin [options] [shorthand ...]
```

Runs on:

QNX Neutrino

Options:

-d *delay*

The delay, in tenths of a second, to use when looping with the -l option. The default is 10.

-F *formats*

A combination of format codes, like the format string for *printf()*. Each code consists of a percent sign (%), an optional width for the field, and a format character (e.g., `"%I %60N"`). For more information, see "*Format characters* (p. 1524)," below.

If you don't specify any format codes, the default is `"%a %b %N %p %J %B"`.

-f *formats*

The same as the -F option, but the *formats* parameter is a contiguous string of format codes that gets expanded. For example, -f mbe is expanded to -F "%m %b %e".

-h

Display a brief usage message.

-k

Keep displaying data for PIDs and TIDs until an error occurs, for example, encountering a PID/TID in an unknown state (because the PID/TID is partially alive).

-l

Copyright © 2014, QNX Software Systems Limited

Loop mode; display statistics every *delay* tenths of a second (specified with the -d option).

-M ***formats***

A combination of format characters, each following a percent sign (%), like the format string for *printf()*, that controls the formatting of information about memory regions. For more information, see "*Memory format characters* (p. 1533)," below.

-n ***node***

The name of the remote node from which to get the information.

The -n option isn't compatible with the o format or the extsched and fds shorthands. Instead, you can use *on -f* (p. 1417); for example:

```
on -f remote_node pidin fds
```

-o ***prio***

Run at *prio* priority.

-P ***pid***

Show only the process family you're interested in (*pid* may be a name or number).

-p ***pid***

Show only the process you're interested in (*pid* may be a name or number).

If the *pid* for the -P or -p option is a number, it's interpreted as a process ID; otherwise, it's interpreted as a name. To avoid confusion, don't assign a numerical name to a process.

shorthand

A name that represents a certain combination of format codes or a special command:

- arguments
- backtrace
- channels
- environment
- extsched

Copyright © 2014, QNX Software Systems Limited

- family
- fds
- flags
- info
- irqs
- libs
- mapinfo
- memory
- net
- pmem
- rc
- regs
- rmasks
- sched
- session
- signals
- syspage
- threads
- timers
- times
- tolerance
- ttimes
- users

You need to type only as many characters of the name as are required to uniquely identify it. For more information, see "*Shorthand forms* (p. 1534)," below.

Description:

The `pidin` utility displays information about the processes running on a QNX Neutrino system.

- This utility goes through all of the processes in the system, one by one, and retrieves state information for each; it doesn't get the system state in a single snapshot. Thus `pidin`'s output reflects what was happening at the instant in time when it queried each process, so you might see some anomalies, suchs as more threads running than there are processors in the system.
- If you aren't logged in as `root`, `pidin` can't get full information about processes owned by other users.

Copyright © 2014, QNX Software Systems Limited

By default, `pidin` displays the statistics once and then exits. If you specify the -l option, `pidin` loops forever, displaying statistics after the delay specified by the -d option.

If you specify the -l and -k options, `pidin` loops until a error occurs, displaying statistics after the given delay. The most common error encountered is a race condition: `procnto` indicates that a process exists, but the process is gone when `pidin` queries it.

Format characters

The format characters that you can specify with the -F or -f option include the following:

A

Display the arguments.

a

Display the process ID.

B

Display what you're blocked on; The output includes a `Blocked` column whose value depends on the thread's state:

State	Value
`CONDVAR`	Address of the condvar
`JOIN`	Thread ID of the blocking thread
`MUTEX`	The address of the mutex, or the IDs of the process and thread blocked on, followed by the number of times locked, in the form *pid-tid #times*
`RECEIVE`	ID of the channel within the process that the thread is blocked on
`REPLY`	Process ID[a]
`SEM`	Address of the semaphore
`SEND`	Process ID[a]
`STACK`	Stack size
`WAITPAGE`	Virtual address of the page
`WAITTHREAD`	Thread ID of the blocking thread

Copyright © 2014, QNX Software Systems Limited

[a] If the process is running on a remote node, the process ID is followed by @ and the node name.

b

Display the thread ID.

C

Display the process ID of one of the process's children. If you specify the `G` format, other children are listed as siblings of this child.

c

Display the code size of the process.

D

Display the process's debug flags:

Flag	Value	Meaning
_DEBUG_FLAG_STOPPED	0x00000001	The thread isn't running.
_DEBUG_FLAG_ISTOP	0x00000002	The thread is stopped at apoint of interest.
_DEBUG_FLAG_IPINVAL	0x00000010	The instruction pointer isn't valid.
_DEBUG_FLAG_ISSYS	0x00000020	System process.
_DEBUG_FLAG_SSTEP	0x00000040	Stopped because of single-stepping.
_DEBUG_FLAG_CURTID	0x00000080	The thread is the current thread.
_DEBUG_FLAG_TRACE_EXEC	0x00000100	Stopped because of a breakpoint.
_DEBUG_FLAG_TRACE_RD	0x00000200	Stopped because of read access.
_DEBUG_FLAG_TRACE_WR	0x00000400	Stopped because of write access.
_DEBUG_FLAG_TRACE_MODIFY	0x00000800	Stopped because of modified memory.
_DEBUG_FLAG_RLC	0x00010000	The Run-on-Last-Close flag is set.

Copyright © 2014, QNX Software Systems Limited

Flag	Value	Meaning
_DEBUG_FLAG_KLC	0x00020000	The Kill-on-Last-Close flag is set.
_DEBUG_FLAG_FORK	0x00040000	The child inherits flags (stop on fork or spawn).

d

Display the data size of the process.

E

Display the environment.

e

Display the parent PID.

F

Show the threads' flags in hexadecimal, as follows:

Flag	Value	Meaning
_NTO_TF_INTR_PENDING	0x00010000	The thread has a pending interrupt
_NTO_TF_DETACHED	0x00020000	The thread is detached
_NTO_TF_THREADS_HOLD	0x00100000	Threads are being held
_NTO_TF_UNBLOCK_REQ	0x00400000	There's an unblock pending on the thread
_NTO_TF_ALIGN_FAULT	0x01000000	An alignment fault has occurred
_NTO_TF_SSTEP	0x02000000	Single-stepping is turned on
_NTO_TF_ALLOCED_STACK	0x04000000	Memory used in the stack isn't automatically returned to the system heap when the thread exits
_NTO_TF_NOMULTISIG	0x08000000	Signals don't terminate all threads in the process

Copyright © 2014, QNX Software Systems Limited

Flag	Value	Meaning
_NTO_TF_FROZEN	0x10000000	The thread is frozen
_NTO_TF_IOPRIV	0x80000000	The thread has I/O privileges

f

Show the processes' flags in hexadecimal, as follows:

Flag	Value	Meaning
_NTO_PF_NOCLDSTOP	0x00000001	The process isn't sent a SIGCHILD signal when its children stop
_NTO_PF_LOADING	0x00000002	The process hasn't been fully loaded
_NTO_PF_TERMING	0x00000004	The process is terminating
_NTO_PF_ZOMBIE	0x00000008	The process is a zombie
_NTO_PF_NOZOMBIE	0x00000010	The process won't become a zombie on its death
_NTO_PF_FORKED	0x00000020	The process is a child by way of *exec()*
_NTO_PF_ORPHAN_PGRP	0x00000040	The process is an orphan
_NTO_PF_STOPPED	0x00000080	The process has been stopped
_NTO_PF_DEBUG_STOPPED	0x00000100	The process has been stopped by the debugger
_NTO_PF_BKGND_PGRP	0x00000200	The process is running in the background
_NTO_PF_NO_LIMITS	0x00000400	The process has no limits on its resources
_NTO_PF_CONTINUED	0x00000800	The process was stopped, but has now been made to continue

Copyright © 2014, QNX Software Systems Limited

Flag	Value	Meaning
_NTO_PF_CHECK_INTR	0x00001000	The process is attached to some interrupts
_NTO_PF_COREDUMP	0x00002000	The process has written a coredump file
_NTO_PF_PTRACED	0x00004000	The process is being traced
_NTO_PF_RING0	0x00008000	The process is running in a privileged supervisor state (known as “ring 0” in some architectures)
_NTO_PF_SLEADER	0x00010000	The process is a session leader
_NTO_PF_WAITINFO	0x00020000	The process will produce wait information when it terminates
_NTO_PF_VFORKED	0x00040000	The process was created with *vfork()*
_NTO_PF_DESTROYALL	0x00080000	The process is being destroyed
_NTO_PF_NOCOREDUMP	0x00100000	The process is not permitted to create core files
_NTO_PF_NOCTTY	0x00200000	The process doesn't have a controlling TTY
_NTO_PF_THREADWATCH	0x80000000	Not currently used

G

Display the process ID of one of the process's siblings. If you specify the `C` format, `pidin` displays the process ID of one of each process's children. You can use the child and sibling information to determine a process's children.

H

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino Core OS 6.3.2 or later) Display the scheduling-specific information for each thread.

For adaptive partitioning scheduling, the information is the name of the partition that the thread is running in. For more information, see the Adaptive Partitioning *User's Guide*.

h

(QNX Neutrino Core OS 6.3.2 or later) Display the thread name; if a thread doesn't have a name, `pidin` displays the thread's ID (*tid*) instead.

I

Display the PID and TID, joined by a hyphen.

i

(QNX Neutrino Core OS 6.3.2 or later) Display the runmask and inherit mask. For more information, see the *Multicore Processing* User's Guide.

J

Display the state of the thread; see "Thread life cycle" in the QNX Neutrino Microkernel chapter of the *System Architecture* guide

K

Display the last kernel call that was executed.

L

Display the session ID.

l

("el") Display the last CPU that the thread ran on.

M

Display the memory owned by the PID.

m

Display the stack size of the process. A * next to a stack size indicates that memory used in the stack isn't automatically returned to the system heap when the thread exits. The memory is returned when the process exits.

N

Display the short name of the process.

n

Copyright © 2014, QNX Software Systems Limited

Display the long name of the process.

O

(QNX Neutrino 6.6 or later) Display the loaded shared libraries.

o

(QNX Neutrino Core OS 6.3.2 or later) Display the connection IDs and file descriptors associated with the process. If you don't have permission to access them, `pidin` provides only limited information.

The information for each connection and file descriptor includes the following:

- the file descriptor, followed by `s` if it's a side channel
- the ID of the process the connection is to
- open flags (`r` or `-`, followed by `w` or `-`), or `MP` for a mountpoint
- the offset
- the name of the file or device, if available.

The -n option isn't compatible with the `o` format; use `on -f` *`remote_node`* `pidin -F "%o"` instead.

P

(Uppercase “P”) Display the parent group.

p

(Lowercase “p”) Display the thread priority. The letter following the scheduling priority number stands for the scheduling policy used, as follows:

- `f` — FIFO scheduling
- `r` — round-robin scheduling
- `o` — other (currently the same as round-robin scheduling)
- `s` — sporadic scheduling

For more information on these scheduling policies, see “Thread scheduling” in the *System Architecture* guide.

Q

Display the interrupt handlers. For each handler, `pidin` shows:

- the interrupt ID returned by *InterruptAttach()* or *InterruptAttachEvent()*
- the interrupt vector passed to *InterruptAttach()* or *InterruptAttachEvent()*
- the mask level count

Copyright © 2014, QNX Software Systems Limited

- the interrupt flags. Valid flags include:
  - T—_NTO_INTR_FLAGS_TRK_MSK
  - P—_NTO_INTR_FLAGS_PROCESS
  - E—_NTO_INTR_FLAGS_END
- the address of the interrupt handler and the interrupt handler area, separated by a colon
- a description of the sigevent to be delivered. Valid descriptions include:
  - SIGNAL *signo*
  - SIGNAL_CODE *signo code*:*value*
  - SIGNAL_THREAD *signo code*:*value*
  - PULSE *coid*:*priority code*:*value*<
  - UNBLOCK
  - INTR
  - THREAD *code*:*value*

q

(QNX Neutrino 6.4.0 or later) Display the backtrace of the addresses of calling routines. For best results, use this format with the `I` format (as in the `backtrace` shorthand form).

R

Display information about the timers:

- the timer ID returned by *TimerCreate()*
- the thread ID associated with this timer (0 for the entire process)
- the number of overruns
- the type of clock used:
  - REAL — CLOCK_REALTIME
  - SOFT — CLOCK_SOFTTIME
  - MONO — CLOCK_MONOTONIC
- the timer flags:
  - X — _NTO_TI_EXPIRED
  - A — _NTO_TI_ABSOLUTE
  - a — _NTO_TI_ACTIVE
  - P — _NTO_TI_PRECISE
- the time left before expiry, in microseconds, followed by a slash (/), and then by the timer interval, in microseconds

Copyright © 2014, QNX Software Systems Limited

- the timer tolerance, in microseconds, or `INF` for infinitely tolerant timers
- the description of the `sigevent` to be delivered when the timer expires:
  - SIGNAL *signo*
  - SIGNAL_CODE *signo code*:*value*
  - SIGNAL_THREAD *signo code*:*value*
  - PULSE *coid*:*priority code*:*value*<
  - UNBLOCK
  - INTR
  - THREAD *code*:*value*

r

Show the values of the registers.

S

Display the signal-ignore mask.

s

Display the signal-queued mask.

T

Display the number of threads.

t

Display the time at which the process was started.

U

Display the process's user ID, as a number.

u

Display the number of nanoseconds spent running in user space.

V

Display the process's group ID, as a number.

v

Display the number of nanoseconds spent running in system space.

W

Display the process's effective user ID, as a number.

w

Copyright © 2014, QNX Software Systems Limited

Display the number of nanoseconds that the process's terminated children spent running in user space.

X

Display the process's effective group ID, as a number.

x

Display the number of nanoseconds that the process's terminated children spent running in system space.

Y

Display the process's set user ID, as a number.

y

Display the time at which the thread was started.

Z

Display the process's set group ID, as a number.

z

Display the number of nanoseconds that the thread spent running in user and system space.

[

(QNX Neutrino 6.4.0 or later) Display the lengths of the send, receive, reply and pulse queues.

****

(QNX Neutrino 6.6 or later) Display the process's supplementary group IDs.

]

(QNX Neutrino 6.6 or later) Display the process's application ID.

^

(QNX Neutrino 6.6 or later) Display the process's default timer tolerance in nanoseconds, or `infinite` if the timers are infinitely tolerant.

Memory format characters

The memory format characters that you can specify with the -M options include the following:

<

Display the memory object's code size.

Copyright © 2014, QNX Software Systems Limited

=

Display the memory object's data size.

>

Display the memory object's address.

?

Display the memory object's offset.

M

Display the memory owned by the process.

: (colon)

Display the memory object's name, or `Mapped Phys Memory` for mapped physical memory.

; (semicolon)

Display the offset that was used in the *mmap()* call when physical memory was mapped.

@

Display the memory object's flags, which can include:

- ANON — MAP_PRIVATE, MAP_ANON
- E — MAP_ELF
- F — MAP_FIXED
- P — MAP_PRIVATE
- S — MAP_SHARED

For more information about these flags, see the entry for *mmap()* in the QNX Neutrino *Library Reference*.

Shorthand forms

Each shorthand form represents a certain combination of format codes or a special command. You need to type only as many characters of the name as are required to uniquely identify it. The shorthand forms include the following:

arguments (equivalent to -F "%a %A")

Show the arguments of the displayed processes.

backtrace (equivalent to -F "%I %q")

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino 6.4.0 or later) Display backtrace information for each thread in the displayed processes. For example:

```
$ pidin -p devc-con-hid back
pid-tid     backtrace
    4103-01 b033ab5b:b03323cb:b03324f3:804f6ed:804c120:804a285
    4103-02 b033af63:805ca60:b031f0ad
```

The output includes the process ID hyphenated to the thread ID, followed by a backtrace of the addresses of the calling routines.

channels (equivalent to -F "%a %b %N %[")

(QNX Neutrino 6.4.0 or later) Display the lengths of the send, receive, reply and pulse queues.

This shorthand is useful if you're trying to track pulse leaks—that is, a process not receiving pulses. This can cause a growth in kernel memory usage, since pulse structures are allocated in the kernel.

environment (equivalent to -F "%a %N %E")

Show the environment of the displayed processes.

extsched

(QNX Neutrino Core OS 6.3.2 or later) Display details of the active extended scheduler configuration.

For the adaptive partitioning scheduler, this is one line of global configuration and then one line for each defined partition (showing the name, budget, critical budget, and overload notifications). For more information, see the Adaptive Partitioning *User's Guide*.

The extsched shorthand is supported only for the local node. To get this information from a remote node, use:

```
on -f remote_node pidin extsched
```

family (equivalent to -F "%a %N %L %P %e %G %C")

Show the sessions, process groups, parents, siblings, and children of the displayed processes.

fds (equivalent to -F "%a %N %o")

(QNX Neutrino Core OS 6.3.2 or later) Show information about the process's connections and file descriptors.

The -n option isn't compatible with the fds shorthand; use `on -f remote_node pidin fds` instead.

Copyright © 2014, QNX Software Systems Limited

flags (equivalent to -F "%a %N %f")

Show the processes' flags in hexadecimal.

info

Display information about the system, such as the type of processor(s) and the amount of free memory.

irqs (equivalent to -F "%a %b %N %Q")

Show the IRQ handlers owned by the process.

libs (equivalent to -F "%a %N %O")

(QNX Neutrino 6.6 or later) Show the shared libraries loaded by the process.

mapinfo (equivalent to -F "%a %b %N %p %J %c %d %m" -M " %: @%> %; %< %= %@")

(QNX Neutrino 6.4.0 or later) Show information about memory mappings.

The output looks like this:

```
4101   8 proc/boot/io-usb    10o RECEIVE          80K  424K   4096(20K)
         libc.so.3          @b0300000           452K   16K
         devu-uhci.so       @b8200000            24K  4096
         devu-ohci.so       @b8207000            24K  4096
         devu-ehci.so       @b820e000            28K  4096
         Mapped Phys Memory @40100000 (ee000000)         12K  S
```

It includes:

- the memory object's name, or `Mapped Phys Memory` for mapped physical memory
- the memory object's address, followed by the offset if applicable
- the object's code and data sizes
- the memory object's flags

memory (equivalent to -F "%a %b %N %p %J %c %d %m" -M " %M @%> %? %< %=")

Show the memory used by the displayed processes; `pidin` displays the shared memory regions, including shared objects, and stack usage for each thread. Shared code and data regions are removed from the size of the process.

The stack numbers represent the amount of stack currently mapped and, in brackets, the maximum allowed for that process.

A * next to a stack size indicates that memory used in the stack isn't automatically returned to the system heap when the thread exits. The memory is returned when the process exits.

Copyright © 2014, QNX Software Systems Limited

Entries for `/dev/mem` indicate shared memory that's mapped into the process address space. For example:

```
/dev/mem          @38100000 (       0)        172K
```

If the entries for different processes show the same object (`@38100000` in this example), they all reference the same shared memory object. The processes can map that shared memory differently; the number in parentheses is the offset that was used in the *mmap()* call, and the last number is the size of the mapping.

If a shared object that contains text relocations is remapped as private, `pidin mem` displays an exclamation mark (`!`) beside the name.

net

Display system information about all the nodes on the Qnet network.

pmem (equivalent to -F "%a %b %N %p %J %c %d %m")

Display process memory only.

rc

Show the process name and arguments of all remote nodes connected to your machine.

regs (equivalent to -F "%a %b %N %r")

Show the values of the registers.

rmasks (equivalent to -F "%a %b %N %i")

(QNX Neutrino Core OS 6.3.2 or later) Display runmasks and inherit masks.

sched (equivalent to -F "%a %b %N %p %l %H %J")

(QNX Neutrino Core OS 6.3.2 or later) Display useful scheduling parameters for each thread.

session (equivalent to -F "%L %a %P %e %N")

Sort by session ID, then process ID. By default, `pidin` sorts the output by process ID.

signals (equivalent to -F "%a %b %N %S %s")

Show the signal state of the displayed processes.

syspage

Copyright © 2014, QNX Software Systems Limited

Show the `syspage` entry. You can specify which section to print by indicating a name (e.g. the command `pidin syspage=asinfo` displays the `asinfo` section). The default is all.

For example, if you want to find out how much space the image file system (IFS) occupies in the memory, run the following command:

```
pidin syspage=asinfo
```

and look for the lines with `imagefs`. See the output in the display as shown in the "Examples:" section below.

For more information, see "Structure of the system page" in the Customizing Image Startup Programs chapter of *Building Embedded Systems*.

threads (equivalent to -F "%a %N %h %J %B")

Show the thread name; if a thread doesn't have a name, `pidin` displays the thread's ID (*tid*) instead.

timers (equivalent to -F "%a %b %N %R")

Show the timers owned by the process.

times (equivalent to -F "%a %N %L %t %u %v %w %x")

Display times for the process.

For each process displayed, show:

- `start time`—the time and date that the process was started
- `utime`—the number of CPU seconds consumed by the process
- `stime`—the number of CPU seconds consumed by the kernel on behalf of the process
- `cutime`—the number of CPU seconds consumed by the children of the process
- `cstime`—the number of CPU seconds consumed by the kernel on behalf of the children of the process

The times for the child processes are added to `cutime` and `cstime` only after the children terminate.

CPU usage is calculated by sampling. When the timer interrupt occurs, the kernel determines which process is running, and adds the time to the total running times of the active thread and its process. If the kernel itself is active, it also adds the time to the system times (`stime`) of the active thread and its process. The `utime` is the total running time minus the system time.

Copyright © 2014, QNX Software Systems Limited

As a result, these times are approximate, and can be inaccurate (e.g., if a process is driven by the timer interrupt). To determine more accurate times, use the system profiler. For more information, see the System Analysis Toolkit *User's Guide*, or the Analyzing Your System with Kernel Tracing chapter of the IDE *User's Guide*.

---

tolerance (equivalent to -F "%a %N %^")

(QNX Neutrino 6.6 or later) Show the process's default timer tolerance.

ttimes (equivalent to -F "%a %b %N %J %t %y %z")

Show thread times.

users (equivalent to -F "%a %N %U %V %W %X %Y %Z %\\")

Display the real, effective, and saved user IDs and group IDs, and the supplemental group IDs for the user who launched the processes.

This option doesn't display the users' or groups' names, just the numerical IDs.

Examples:

The `pidin` command prints a listing similar to this:

```
    pid tid name               prio STATE       Blocked
      1   1 /sys/procnto-instr   0f READY
      1   3 /sys/procnto-instr  10r RUNNING
      1   4 /sys/procnto-instr  12r RECEIVE     1
      1   5 /sys/procnto-instr  12r RECEIVE     1
      1   6 /sys/procnto-instr  12r RECEIVE     1
      1  11 /sys/procnto-instr  12r RECEIVE     1
      1  12 /sys/procnto-instr  10r RECEIVE     1
      1  13 /sys/procnto-instr  10r RECEIVE     1
      1  15 /sys/procnto-instr 255r RECEIVE     1
      1  16 /sys/procnto-instr  10r RECEIVE     1
      1  17 /sys/procnto-instr  10r RECEIVE     1
      2   1 sbin/tinit          10o REPLY       1
      3   1 proc/boot/slogger   10o RECEIVE     1
      5   1 proc/boot/pci-bios  10o RECEIVE     1
      6   1 roc/boot/devb-eide  10o SIGWAITINFO
      6   2 roc/boot/devb-eide  21r RECEIVE     1
      ...
```

Using `pidin -F "%I %60N"` displays the PID and TID, along with up to 60 characters of the processes' short name:

```
pid-tid     name
       1-01 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-03 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-04 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-05 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-06 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-11 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-12 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-13 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-15 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-16 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       1-17 rldbuild/cdr/qnx6/tmp/target/qnx6/x86/boot/sys/procnto-instr
       2-01 sbin/tinit
       3-01 proc/boot/slogger
       5-01 proc/boot/pci-bios
       6-01 proc/boot/devb-eide
       ...
```

The `pidin mem` command displays:

```
    pid tid name               prio STATE          code  data        stack
      1   1 /sys/procnto-instr   0f READY          1812K   12K     0(320)*
```

Copyright © 2014, QNX Software Systems Limited

```
1   3 /sys/procnto-instr  10r RUNNING         1812K   12K     0(8192)
1   4 /sys/procnto-instr  12r RECEIVE         1812K   12K     0(8192)
1   5 /sys/procnto-instr  12r RECEIVE         1812K   12K     0(8192)
1   6 /sys/procnto-instr  12r RECEIVE         1812K   12K     0(8192)
1  11 /sys/procnto-instr  12r RECEIVE         1812K   12K     0(8192)
1  12 /sys/procnto-instr  10r RECEIVE         1812K   12K     0(8192)
1  13 /sys/procnto-instr  10r RECEIVE         1812K   12K     0(8192)
1  15 /sys/procnto-instr 255r RECEIVE         1812K   12K     0(8192)
1  16 /sys/procnto-instr  10r RECEIVE         1812K   12K     0(8192)
1  17 /sys/procnto-instr  10r RECEIVE         1812K   12K     0(8192)
     procnto-instr      @cfbe5000             12K   12K
2   1 sbin/tinit          10o REPLY            8192   36K  4096(516K)*
     ldqnx.so.2         @b0300000            344K   16K
3   1 proc/boot/slogger   10o RECEIVE          8192  104K  4096(516K)*
     ldqnx.so.2         @b0300000            344K   16K
5   1 proc/boot/pci-bios  10o RECEIVE            36K   40K  8192(516K)*
     ldqnx.so.2         @b0300000            344K   16K
6   1 roc/boot/devb-eide  10o SIGWAITINFO        52K   91M  8192(516K)*
6   2 roc/boot/devb-eide  21r RECEIVE            52K   91M   4096(12K)
...
```

The `pidin syspage=asinfo` command displays:

```
Section:asinfo offset:0x00000568 size:0x00000240
    0)               0-ffff             o:ffff a:0000 p:100 n:io
   20)               0-ffffffff         o:ffff a:0010 p:100 n:memory
   40)               0-ffffff           o:0020 a:0010 p:100 n:memory/isa
   a0)               0-9fbff            o:0040 a:0017 p:100 n:memory/isa/ram
  180)            1000-cfff             o:00a0 a:0007 p:100 n:memory/isa/ram/sysram
  1a0)           20f98-9fbff            o:00a0 a:0007 p:100 n:memory/isa/ram/sysram
   c0)          100000-ffffff           o:0040 a:0037 p:100 n:memory/isa/ram
  1c0)          100000-40e507           o:00c0 a:0007 p:100 n:memory/isa/ram/sysram
  1e0)          5e533c-ffffff           o:00c0 a:0027 p:100 n:memory/isa/ram/sysram
   60)         6000000-ffefffff         o:0020 a:0013 p:100 n:memory/device
  100)         6000000-ffeafff          o:0060 a:0017 p:100 n:memory/device/ram
  220)         6000000-ffeafff          o:0100 a:0007 p:100 n:memory/device/ram/sysram
   80)        fff00000-ffffffff         o:0020 a:0005 p:100 n:memory/rom
   e0)         1000000-5ffffff          o:0020 a:0037 p:100 n:memory/ram
  200)         1000000-5ffffff          o:00e0 a:0027 p:100 n:memory/ram/sysram
  120)          40e508-5e533b           o:0020 a:0005 p:100 n:memory/imagefs
  140)          400400-40e507           o:0020 a:0007 p:100 n:memory/startup
  160)          40e508-5e533b           o:0020 a:0007 p:100 n:memory/bootram
```


Copyright © 2014, QNX Software Systems Limited

pin

Display PC Card information

You must be `root` to run this utility.

Syntax:

```
pin [-s socket] [command]
```

Runs on:

QNX Neutrino

Options:

-s *socket*

Display information about this PC Card socket only (starting at 1).

Description:

The `pin` utility displays useful information about PC Card resources, servers and cards.

The *command* specifies the type of information to be displayed:

`status`

Show the status of each socket. This is the default command.

`config`

Attempt to make configuration file entries (obsolete).

`cis`

Decode the Card Information Structure (CIS) on the card.

The QNX Neutrino PC Card server (*`devp-pccard`* (p. 455)) automatically configures PCMCIA/CardBus I/O cards as they're inserted. It also manages computer resources such as memory windows, ports and IRQs. Resources are allocated as cards are inserted and then freed as cards are removed.

Drivers that are PC Card-aware can be informed of card insertions and removals, based on the type of card. These drivers can then configure themselves to use the correct ports and IRQs (see the *pccard_** functions in the QNX Neutrino *C Library Reference*). The *`pccard-launch`* (p. 1451) utility automatically starts drivers and passes them

Copyright © 2014, QNX Software Systems Limited

command-line arguments describing the ports and IRQs used by the card, thus permitting a non-PC Card-aware driver to be automatically started when the card is inserted and stopped when removed.

Copyright © 2014, QNX Software Systems Limited

ping

Send ICMP ECHO_REQUEST packets to network hosts (UNIX)

Syntax:

```
ping [-aDdfLnoPQqRrv] [-c count] [-E policy] [-g gateway]
     [-h host] [-I ifaddr] [-i wait] [-i interval]
     [-l preload] [-p pattern] [-s packetsize] [-t tos]
     [-T ttl] [-w maxwait] host
```

Runs on:

QNX Neutrino

Options:

-a

Emit an audible beep (by sending an ASCII BEL character to the standard error output) after receiving each non-duplicate response. For the sake of your sanity, this option is disabled if you use the -f option to do a flood ping.

-c *count*

Stop after sending (and receiving) this many ECHO_RESPONSE packets.

-D

Set the `Don't Fragment` bit in the IP header. This is meant to determine the path MTU.

-d

Set the SO_DEBUG option on the socket being used.

-E *policy*

Specify the IPsec policy for packets.

-f

Do a “flood ping”: output packets either as fast as they come back or one hundred times per second, whichever is faster.

For every ECHO_REQUEST sent, a “.” is printed; for every ECHO_REPLY received, a backspace is printed. This quickly shows how many packets are being dropped.

Copyright © 2014, QNX Software Systems Limited

Only the superuser (`root`) may use the -f option; it can be very hard on a network — use it with caution.

-g *gateway*

Use Loose Source Routing to send the ECHO_REQUEST packets via *gateway*. The default is to use the routing table.

-h *host*

Alternate way of specifying the target host instead of as the last argument.

-I *ifaddr*

Send multicast datagrams on the network interface specified by the interface's hostname or IP address.

-i *interval*

Wait *interval* seconds between sending each packet (default is one second). For the -f option, the *interval* is 0.01 seconds.

-l *preload*

Send this many packets as fast as possible before returning to normal behavior. Only the superuser may use this option.

-L

Disable loopback when sending to multicast destinations, so the transmitting host doesn't see the ICMP requests.

-n

Print numeric output only. No attempt is made to look up symbolic names for host addresses.

-o

Exit successfully after receiving one reply packet.

-P

Use a pseudo-random sequence for the data instead of the default, fixed sequence of incrementing 8-bit integers. This is useful to foil compression on PPP and other links.

-p *pattern*

Copyright © 2014, QNX Software Systems Limited

Fill out the packet with this many “padding” bytes (maximum is 16). You should find this useful for diagnosing data-dependent problems in a network. For example, -p ff causes the sent packet to be filled with ones.

-Q

Don't display responses such as Network Unreachable ICMP messages concerning the ECHO_REQUESTs sent.

-q

Be quiet: display nothing except for the summary lines at startup time and when finished.

-R

Record the route.

-r

Bypass the normal routing tables and send directly to a host on an attached network. If the host isn't on a directly attached network, an error is returned. You can use this option to ping a local host through an interface that has no route through it (e.g. after the interface was dropped by `routed`).

-s *packetsize*

Send this many data bytes. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

-T *ttl*

Use the specified time-to-live. It represents how many hops the packet can go through before being discarded (when it reaches 0). The default is 255.

-t *tos*

Use the specified hexadecimal type of service.

-v

Verbosity (default none).

-w *maxwait*

Specify a timeout, in seconds, before `ping` exits regardless of how many packets have been sent or received.

Description:

The `ping` utility uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from the given *host* or gateway. ECHO_REQUEST

Copyright © 2014, QNX Software Systems Limited

datagrams, known as *pings*, have an IP and ICMP header, followed by a `timeval` structure and then an arbitrary number of padding bytes used to fill out the packet.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

When using `ping` for isolating faults, you should first run it on the local host to verify that the local network interface is up and running. You should then `ping` hosts and gateways further and further away. Round-trip times and packet-loss statistics are computed. If duplicate packets are received, they aren't included in the packet-loss calculation, although the round-trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets has been sent (and received), or if you terminate `ping` with a SIGINT, a brief summary is displayed.

The `ping` utility is intended for testing, measuring, and managing networks. Because of the load it can impose on the network, you shouldn't use `ping` during normal operations or from automated scripts.

Debugging

You can use the `ping` utility to determine if you have connectivity to other hosts. Suppose you've configured a point-to-point link (PPP), but you haven't specified a default route. You can type the following to see if you're connected to the other end of the link:

```
ping isp.com
```

With success, `ping` outputs something like this:

```
PING isp.com (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=255 time=0 ms
```

This report continues until `ping` is terminated. To terminate `ping`, press **Ctrl**–**C**. You'll see a report like this:

```
--- isp.com ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
```

The `ping` utility may fail for different reasons:

Copyright © 2014, QNX Software Systems Limited

- If nothing is displayed, `ping` may be having problems resolving the hostname. Try the IP address directly to bypass the resolver (see the */etc/nsswitch.conf* (p. 1369) file).
- If only the first line of the above successful output is displayed, then `ping` isn't receiving a response from the specified host.
- All other problems and errors result in an obvious message (e.g. “No route to host”).

ICMP packet details

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. (Specifying a *packetsize* via option -s determines the size of this extra piece of data; default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY is always 8 bytes more than the requested data space (the ICMP header).

If the size of the data space is at least 8 bytes, `ping` uses the first 8 bytes of this space to include a timestamp that it uses in the computation of round-trip times. If less than eight bytes of padding are specified (option -p), no round-trip times are given.

Duplicate and damaged packets

The `ping` utility reports duplicate and damaged packets.

Although they should never happen, duplicate packets can occur in many situations and seem to be caused by inappropriate link-level retransmissions. While duplicates are rarely (if ever) a good sign, the presence of low levels of duplicates isn't always cause for alarm.

Damaged packets, on the other hand, are serious and often indicate malfunctioning hardware somewhere in the ping packet's path (in the network or in the hosts).

Trying different data patterns

The (inter)network layer should never treat packets according to the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases, the particular pattern that will have problems is something that doesn't have sufficient “transitions,” such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros, for example, on the command line because the pattern of interest is at the data-link level—the relationship between what you type and what the controllers transmit can be complicated.

So if you have a data-dependent problem, you'll probably have to do a lot of testing to find it. If you're lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the -p option.

TTL details

Copyright © 2014, QNX Software Systems Limited

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one. The TCP/IP specification states that the TTL field for TCP packets should be set to `60`, but many systems use smaller values (4.3 BSD uses `30`, 4.2 uses `15`).

The maximum possible value of this field is 255, and most UNIX systems (including QNX Neutrino) set the TTL field of ICMP ECHO_REQUEST packets to 255. Thus you'll find you can "ping" some hosts, but not reach them with `telnet` or `ftp`.

In normal operation, `ping` prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

- *Not change it* — this is what Berkeley UNIX systems did before the 4.3BSD-Tahoe release. The TTL value in the received packet is 255 minus the number of routers in the round-trip path.
- *Set it to 255* — this is what current Berkeley UNIX systems do. The TTL value in the received packet is 255 minus the number of routers in the path from the remote system to the pinging host.
- *Set it to some other value* — some machines use the same value for ICMP packets that they use for TCP packets; for example, either `30` or `60`. Others may use completely wild values.

Files:

The `ping` utility requires the `libsocket.so` shared library.

Exit status:

0

Success (the host is alive).

Nonzero

An error occurred. The arguments are incorrect or the host isn't responding.

Copyright © 2014, QNX Software Systems Limited

ping6

Send ICMPv6 ECHO_REQUEST packets to network hosts (UNIX)

Syntax:

```
ping6 [-dfHnNqRvw] [-a addrtype] [-b bufsiz] [-c count]
      [-h hoplimit] [-I interface] [-i wait] [-l preload]
      [-p pattern] [-P policy] [-S sourceaddr] [-s packetsize]

      [hops...] host
```

Runs on:

QNX Neutrino

Options:

-a *addrtype*

Generate an ICMPv6 Node Information Node Addresses query (rather than ECHO_RESPONSE), where *addrtype* is a string constructed from the following characters:

A

Request that the responder's anycast addresses be returned (if not specified, return unicast addresses only). Note that the specification doesn't specify how to get responder's anycast addresses. This is an experimental option.

a

Request all of the responder's unicast addresses. If omitted, only the addresses that belong to the interface which has the responder's address are requests.

c

Request the responder's IPv4-compatible and IPv4-mapped addresses.

g

Request the responder's global-scope addresses.

l

Request the responder's link-local addresses.

Copyright © 2014, QNX Software Systems Limited

s

Request the responder's site-local addresses.

-b *bufsiz*

Set the socket buffer size.

-c *count*

Stop after sending (and receiving) this many ECHO_RESPONSE packets.

-d

Set the SO_DEBUG option on the socket being used.

-f

Do a "flood ping": output packets either as fast as they come back or one hundred times per second, whichever is faster. For every ECHO_REQUEST sent, a "." is printed; for every ECHO_REPLY received a backspace is printed. This quickly shows how many packets are being dropped.

Only the superuser (`root`) may use the -f option; it can be very hard on a network — use it with caution.

-H

Try a reverse-lookup of the IPv6 addresses.

-h *hoplimit*

Set the IPv6 hoplimit.

-I *interface*

Source packets with the given *interface* address. This flag applies if the ping destination is a multicast address, or a link-local/site-local unicast address.

-i *wait*

Wait this many seconds between sending each packet (default is one second). This option is incompatible with -f.

-l *preload*

Send this many packets as fast as possible before returning to normal behavior. Only the superuser (`root`) may use this option.

-n

Copyright © 2014, QNX Software Systems Limited

Print numeric output only. No attempt is made to lookup symbolic names for host addresses.

-N

Probe the node information multicast group (`ff02::2:`*xxxx*`:`*xxxx*). The *host* argument must be a string hostname of the target (it can't be a numeric IPv6 address). The node information multicast group is computed based on a given *host*, and is used as the final destination. Since the node information multicast group is a link-local multicast group, the destination link needs to be specified by -I.

-p ***pattern***

Fill out the packet with this many "pad" bytes (maximum is 16). You should find this useful for diagnosing data-dependent problems in a network. For example, -p ff causes the sent packet to be filled with ones. If specified with -q, any ICMP error messages caused by its own ECHO_REQUEST messages are printed.

-P ***policy***

Specify the IPsec policy to use for the probe.

-q

Be quiet: display nothing except the summary lines at startup time and when finished.

-R

Make the kernel believe that the target *host* (or the first hop if you specify *hops*) is reachable by injecting an upper-layer reachability confirmation hint. The option is meaningful only if the target *host* (or the first hop) is a neighbor.

-S ***sourceaddr***

Specify the source address of request packets. The source address must be one of the unicast addresses of the sending node. If the outgoing interface is specified by the -I option as well, *sourceaddr* needs to be an address assigned to the specified interface.

-s ***packetsize***

Send this many data bytes. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data. You may also need to specify -b to extend socket buffer size.

-v

Copyright © 2014, QNX Software Systems Limited

Verbosity. List all ICMP packets received.

-W

Same as -w, but use the old packet format based on 03 draft. This option is present for backward compatibility.

-w

Generate an ICMPv6 Node Information FQDN query, rather than ECHO_REQUEST. The -s option has no effect when -w is specified.

hops

The IPv6 addresses for intermediate nodes, which are put into a type 0 routing header.

host

The IPv6 adddress of the final destination node.

Description:

The `ping6` utility uses the ICMPv6 protocol's mandatory ICMP6_ECHO_REQUEST datagram to elicit an ICMP6_ECHO_RESPONSE from the given *host* or gateway. ICMP6_ECHO_REQUEST datagrams, known as *pings*, have an IPv6 header, and ICMPv6 header formatted as documented in *RFC 2463*.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

When using `ping6` for isolating faults, you should first run it on the local host to verify that the local network interface is up and running. You should then "ping" hosts and gateways further and further away. Roundtrip times and packet-loss statistics are computed. If duplicate packets are received, they aren't included in the packet-loss calculation, although the roundtrip time of these packets is used in calculating the minimum/average/maximum roundtrip time numbers. When the specified number of packets has been sent (and received), or if you terminate `ping6` with a `SIGINT`, a brief summary is displayed.

The `ping6` utility is intended for testing, measuring, and managing networks. Because of the load it can impose on the network, you shouldn't use `ping6` during normal operations or from automated scripts.

Copyright © 2014, QNX Software Systems Limited

Duplicate and damaged packets

The `ping6` utility reports duplicate and damaged packets.

Although they should never happen when pinging a unicast address, duplicate packets can occur in many situations and seem to be caused by inappropriate link-level retransmissions. While duplicates are rarely (if ever) a good sign, the presence of low levels of duplicates isn't always cause for alarm. Duplicates are expected when pinging a broadcast or multicast address, since they're not really duplicates but replies from different hosts to the same request.

Damaged packets, on the other hand, are serious and often indicate malfunctioning hardware somewhere in the `ping6` packet's path (in the network or in the hosts).

Trying different data patterns

The (inter)network layer should never treat packets according to the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases, the particular pattern that will have problems is something that doesn't have sufficient "transitions," such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros, for example, on the command line because the pattern of interest is at the data-link level—the relationship between what you type and what the controllers transmit can be complicated.

So if you have a data-dependent problem, you'll probably have to do a lot of testing to find it. If you're lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the -p option.

Based on:

- A. Conta, and S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, *RFC 2463*, December 1998.
- Matt Crawford, *IPv6 Node Information Queries, draft-ietf-ipngwg-icmp-name-lookups-05.txt*, October 22, 1999, work in progress material.

Files:

The `ping6` utility requires the `libsocket.so` shared library.

Exit status:

0

Success (the host is alive).

Nonzero

Copyright © 2014, QNX Software Systems Limited

An error occurred. The arguments are incorrect or the host isn't responding.

Copyright © 2014, QNX Software Systems Limited

pipe

Pipe manager (QNX Neutrino)

You must be `root` to start this manager.

Syntax:

```
pipe [-1Pd] [-a atomic_write_size] [-s pipe_buffer_size]
     [-U user_name | uid[:gid[,sup_gid]*]]] &
```

Runs on:

QNX Neutrino

Options:

-1

("One"—QNX Neutrino Core OS 6.3.2 or later) Unblock *select()* for writing when `_PC_PIPE_BUF` bytes are available (the default is 1 byte). For more information about `_PC_PIPE_BUF`, see the Description section below.

-a *atomic_write_size*

(QNX Neutrino Core OS 6.3.2 or later) Set the atomic write size for `_PC_PIPE_BUF` (the default is 5120 bytes). For more information about `_PC_PIPE_BUF`, see the Description section below.

-d

(QNX Neutrino Core OS 6.3.2 or later) Don't daemonize.

-P

(QNX Neutrino Core OS 6.3.2 or later) Synchronize the modification of POSIX attributes to the host filesystem.

-s *pipe_buffer_size*

(QNX Neutrino Core OS 6.3.2 or later) Set the total buffer size (the default is 5120 bytes).

-U *user_name*

-U *uid*[:*gid*[,*sup_gid*]*]]

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

Description:

The `pipe` manager implements the subset of file I/O known as pipes (or anonymous FIFOs). A simple form of interprocess communication, pipes are mostly used to connect the output of one process to the input of another to form a series of filters.

This version of `pipe` is backward compatible. When you invoke `pipe` with no arguments, the behavior you get is identical to that of the versions of `pipe` shipped before the QNX Neutrino Core OS 6.3.2.

The *atomic_write_size* is the size for which it's guaranteed that a *write()* of data won't be interleaved with data from any other process; in other words the amount of data that will be written atomically. This is required when there are multiple writers sending data to a single reader. The POSIX maximum is referred to as `PIPE_BUF`, and can be queried at runtime using `fpathconf(_PC_PIPE_BUF)`. It isn't recommended to lower this value from the default 5120 bytes in case application code is using the `PIPE_BUF` manifest from `<limits.h>` rather than a runtime *fpathconf()* determination.

The *pipe_buffer_size* is the total buffer size of the pipe (and must be at least the atomic write size). Effectively the `pipe` utility can hold this many (unread) bytes before *write()* clients start to block or fail with `EAGAIN` (based on their `O_NONBLOCK` orientation).

There is no simple mechanism (such as *fpathconf()*) to query the pipe-buffering capacity, but it can be determined by writing one byte at a time to an empty `O_NONBLOCK` pipe and counting how many are consumed until an `EAGAIN` error (full-pipe indication) is returned. Increasing this value could make pipelines more *elastic*, in terms of decoupling the writer from the reader processes, at the expense of using extra system memory for each pipe. Across a sample of OSs, pipe-buffering capacity is in the 4–16 KB range.

Strict POSIX conformance requires that the *atime* and *mtime* attributes of a FIFO be updated for every *read()* and *write()* operation. As FIFOs operate by redirecting access of the FIFO from the host filesystem to the pipe server, maintaining these attributes requires additional synchronization messages from the pipe server back to the host

Copyright © 2014, QNX Software Systems Limited

filesystem. In most cases, this level of synchronization isn't required, and by default is performed only at the last *close()* of the FIFO. For strict POSIX conformance, specify the -P option.

The -l option sets the behavior for *select()* write notification of the `pipe` utility. The original behavior was to satisfy or unblock *select()* clients even if 1 byte of buffer space was available. POSIX specifies a different behavior. Since the atomic write size and the total buffer size are both equal to 5120 bytes, *select()* would be satisfied only on an empty pipe, which may impact performance or establish a lock-step timing between the writing and reading processes. If this option is enabled, consider also increasing the -s to be larger than `PIPE_BUF` so that the pipe doesn't have to completely drain before a *select()* for write is satisfied.

Exit status:

The `pipe` utility terminates only if an error occurs during startup. If `pipe` fails during startup, it prints a diagnostic message to the standard output stream and then exits with a nonzero status.

Copyright © 2014, QNX Software Systems Limited

plainrsa_gen

Generator for Plain RSA keys

Syntax:

```
plainrsa_gen [-b bits] [-e pubexp] [-f outfile] [-h]
```

Runs on:

QNX Neutrino

Options:

-b *bits*

The bit length of the key. The default is 1024; the recommended length is 2048 or even 4096 bits. Note that generating longer keys takes more time.

-e *pubexp*

The value of the RSA public exponent. The default is `0x3`.

Don't change this unless you really know what you are doing!

-f *outfile*

Write the resulting key to *outfile* instead of to *stdout*. If the file already exists, it isn't overwritten, so that you won't accidentally lose your private key.

-h

Display help.

Description:

You can use `plainrsa_gen` to generate Plain RSA keys for authentication purposes. Using Plain RSA keys is optional. Other possibilities are pre-shared keys or X.509 certificates.

Output file format

This is the secret private key that should never leave your computer:

```
: RSA   {
        # RSA 1024 bits
        # pubkey=0sAQOrWlcwbAIdNSMhDt...
        Modulus: 0xab5a57306c021d3523...
```


Copyright © 2014, QNX Software Systems Limited

```
        PublicExponent: 0x03
        PrivateExponent: 0x723c3a2048...
        Prime1: 0xd309b30e6adf9d85c01...
        Prime2: 0xcfdc2a8aa5b2b3c90e3...
        Exponent1: 0x8cb122099c9513ae...
        Exponent2: 0x8a92c7071921cd30...
        Coefficient: 0x722751305eafe9...
  }
```

The line `pubkey=0sAQOrW...` of the private key contains a public key that should be stored in the other peer's configuration in this format:

```
: PUB 0sAQOrWlcwbAIdNSMhDt...
```

You can also specify from and to addresses for which the key is valid:

```
0.0.0.0/0 10.20.30.0/24 : PUB 0sAQOrWlcwbAIdNSMhDt...
```

Contributing author:

BSD

Copyright © 2014, QNX Software Systems Limited

pppd

Point-to-Point protocol daemon

Syntax:

```
pppd [options]
```

Runs on:

QNX Neutrino

Options:

For details about the options, see *http://netbsd.gw.com/cgi-bin/man-cgi?pppd++NetBSD-4.0* in the NetBSD documentation. The following options are supported by NetBSD but aren't documented there:

+chap

Require the peer to authenticate itself using CHAP (Challenge-Handshake Authentication Protocol) authentication. Default is no authentication required (usually a server option).

netmask *n*

Set the interface netmask to *n*, a 32-bit netmask in "decimal dot" notation (e.g. 255.255.255.0). The default depends on the class of the IP address (usually a server option).

nologfd

Don't send log messages to any file descriptor.

+pap

Require the peer to authenticate itself using PAP (Password Authentication Protocol). The default is no authentication required (usually a server option).

QNX Neutrino supports multilink PPP, so our pppd daemon supports the following options, contrary to what the NetBSD documentation says:

- mp
- mpshortseq
- mrru
- multilink

Copyright © 2014, QNX Software Systems Limited

- nomp
- nompshortseq
- nomultilink
- pass-filter

The following are specific to QNX Neutrino:

confstr

Write the server-supplied nameserver address to the _CS_RESOLVE configuration string (the default).

noconfstr

Don't write the server-supplied nameserver address to the _CS_RESOLVE configuration string.

noresconf

Don't write the server-supplied nameserver address to `/etc/resolv.conf file` (the default).

resconf

Write the server-supplied nameserver address to the `/etc/resolv.conf` file.

+stdinsecret

Read PAP or CHAP secrets from standard input. If you use this option, you need to specify explicitly a serial device on the command line.

usefd ***filedes***

Use this file descriptor to send or receive pppd packets instead of opening a *tty_name*.

useuserdns ***nameserver_IP***

Specify the nameserver to use. This overrides any nameservers provided by the server.

Description:

The pppd daemon is used to establish TCP/IP serial connections using the point-to-point protocol (PPP). For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?pppd++NetBSD-4.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Based on:

RFC 1144, *RFC 1321*, *RFC 1332*, *RFC 1334*, *RFC 1549*, *RFC 1661*, *RFC 1662*, *RFC 1962*, *RFC 1990*

Caveats:

The following signals have the specified effect when sent to the pppd process:

SIGINT, SIGTERM

These signals cause pppd to terminate the link (by closing LCP), restore the serial device settings, and exit.

SIGHUP

Indicates that the physical layer has been disconnected; pppd attempts to restore the serial device settings and then exits.

MS-CHAP

Authentication support is client-side only. It can be used to authenticate ourselves, but not the peer.

If you spawn pppd from another program and specify the nodetach or updetach option, and if a signal is dropped on pppd while it's running a connect or disconnect script, pppd raises the signal on the entire process group, including the parent (i.e. the program that spawned pppd). This could cause the parent to terminate unexpectedly. To avoid this, spawn pppd with the SPAWN_SETGROUP set in the `inheritence` structure. For more information, see *spawn()* in the QNX Neutrino *C Library Reference*.

Copyright © 2014, QNX Software Systems Limited

pppoectl

Display or set parameters for a PPPOE interface

Syntax:

```
pppoectl [-v] ifname [parameter[=value]] [...]
pppoectl -e ethernet-ifname [-s service-name]
         [-a access-concentrator-name] [-d] [-n 1 | 2] ifname
pppoectl -f config-file ifname [...]
```

Runs on:

QNX Neutrino

Options:

-e *ethernet-ifname*

The Ethernet interface to use to communicate with the access concentrator (typically via a DSL modem).

-a *access-concentrator-name*

The name of the access concentrator.

-s *service-name*

The name of the service connected to.

-d

Dump the current connection state information. You typically use this parameter alone, for informational purposes, not when configuring an interface.

-n 1 | 2

Print the IP address of the primary or secondary DNS name server for this PPP connection. This is available only if DNS query is enabled; see the `query-dns` parameter.

-f *config-file*

Parse *config-file*, ignoring lines starting with a #, for *parameter*[=*value*] pairs, one per line, as if they had been specified on the command line. This lets you avoid passing the password as a command-line argument.

Copyright © 2014, QNX Software Systems Limited

Description:

The `pppoectl` utility displays or sets parameters for a `pppoe` interface. There are two basic modes of operation:

- configuring security-related parameters
- attaching a PPPOE interface to its Ethernet interface, optionally passing in additional parameters for the PPPOE encapsulation

The latter usage is indicated by the presence of the -e option, which takes the name of the Ethernet interface as its argument. You don't typically specify both the access concentrator name and the service name.

PPPOE drivers require a number of additional arguments or optional parameters besides the settings that you can adjust with *`ifconfig`* (p. 957). These are things such as authentication protocol parameters, but also other tunable configuration variables. You can use `pppoectl` to display the current settings or adjust these parameters as required.

You always need to specify the *ifname* argument, in order to identify the interface for which to set or display the parameters. Use `ifconfig` or *`netstat`* (p. 1339) to see which interfaces are available.

If no other parameter is given, `pppoectl` just lists the current settings for *ifname*, and then exits. The reported settings include the current PPP phase the interface is in, which can be one of `dead`, `establish`, `authenticate`, `network`, or `terminate`. If an authentication protocol is configured for the interface, the name of the protocol to be used, as well as the system name to be used or expected are displayed, plus any possible options to the authentication protocol if applicable. Note that the authentication secrets (sometimes also called keys) aren't returned by the underlying system call, and thus aren't displayed.

If any additional parameter is supplied, superuser privileges are required, and the command works in “set” mode. This is normally done quietly, unless you specify the -v option, which causes a final printout of the settings as described above once all other actions have been taken. Use of this mode is rejected if the interface is currently in any other phase than dead. Note that you can force an interface into the dead phase by calling *`ifconfig`* (p. 957) with the parameter `down`.

Supported parameters

The currently supported parameters include:

`authproto=`*`protoname`*

Set both his and my authentication protocol to *protoname*. The protocol name can be one of `chap`, `pap`, or `none`. In the latter case, the use of an authentication protocol is turned off for the named interface. This has the side-effect of clearing the other authentication-related parameters for this

Copyright © 2014, QNX Software Systems Limited

interface as well (i.e. the system name and authentication secret will be forgotten).

myauthproto=*protoname*

Same as `authproto`, but only for my end of the link. In other words, this is the protocol when the remote is the authenticator, and I am the peer required to authenticate.

hisauthproto=*protoname*

The same as `authproto`, but only for his end of the link.

myauthname=*name*

Set my system name for the authentication protocol.

hisauthname=*name*

Set his system name for the authentication protocol. For CHAP, this is used only as a hint, causing a warning message if the remote supplied a different name. For PAP, it's the name remote must use to authenticate itself (in connection with its secret).

myauthsecret=*secret*

Set my secret (key, password) for use in the authentication phase. For CHAP, this is used to compute the response hash value, based on remote's challenge. For PAP, it's transmitted as plain text together with the system name. Don't forget to quote the secrets from the shell if they contain shell metacharacters (or white space).

myauthkey=*secret*

The same as `myauthsecret`.

hisauthsecret=*secret*

Similar to `myauthsecret`, to be used if we are the authenticator, and the remote peer needs to authenticate.

hisauthkey=*secret*

The same as `hisauthsecret`.

callin

Require the remote to authenticate itself only when it's calling in, but not when we're the caller. This is required for some peers that don't implement

Copyright © 2014, QNX Software Systems Limited

the authentication protocols symmetrically (such as Ascend routers, for example).

always

The opposite of `callin`. Require the remote to always authenticate, regardless of which side is placing the call. This is the default, and isn't explicitly displayed in `list` mode.

norechallenge

Meaningful only with CHAP. Don't rechallenge a peer once the initial CHAP handshake was successful. Use this to work around broken peer implementations that can't handle being rechallenged once the connection is up.

rechallenge

With CHAP, send rechallenges at random intervals while the connection is in network phase. (The intervals are currently in the range of 300 through approximately 800 seconds.) This is the default, and isn't explicitly displayed in `list` mode.

idle-timeout=*idle-seconds*

For services that are charged by connection time, the interface can optionally disconnect after a configured idle time. If set to 0, this feature is disabled.

For ISDN devices, it's preferable to use the `isdnd`-based timeout mechanism, as `isdnd` can predict the next charging unit for ISDN connections and optimize the timeout with this information.

lcp-timeout=*timeout-value*

Change the value of the LCP timeout. The default value of the LCP timeout is currently 1 second. Specify the *timeout-value* in milliseconds.

max-noreceive=*sec*

Set the number of seconds after the last reception of data from the peer before the line state is probed by sending LCP echo requests. The *sec* interval isn't used verbatim; the first echo request might be delayed up to 10 seconds after the configured interval.

max-alive-missed=*count*

Set the number of unanswered LCP echo requests to tolerate before considering a connection to be dead. LCP echo requests are sent in

Copyright © 2014, QNX Software Systems Limited

10-second intervals after the configured `max-noreceive` interval has passed with no data received from the peer.

`max-auth-failure=`*`count`*

Since some ISPs disable accounts after too many unsuccessful authentication attempts, there is a maximum number of authentication failures before we will stop retrying without manual intervention. Manual intervention is either changing the authentication data (name, password) or setting the maximum retry count. If *count* is set to 0, this feature is disabled.

`clear-auth-failure`

If an authentication failure has been caused by remote problems, and you want to retry connecting using unchanged local settings, you can use this command to reset the failure count to zero.

`query-dns=`*`flags`*

During PPP protocol negotiation, we can query the peer for addresses of two name servers. If *flags* is 1, only the first server address is requested; if *flags* is 2, the second is requested. Setting *flags* to 3 queries both.

You can retrieve the result of the negotiation with the -n option.

If you find `pppoed` has problems connecting to certain sites on the Internet, see the technote PPPOE and Path MTU Discovery.

Examples:

The following example is the complete sequence of commands to bring a PPPOE connection up:

```
# Need ethernet interface UP (or it won't send any packets)
ifconfig ne0 up

# Create the PPPOE interface
ifconfig pppoe0 create

# Let pppoe0 use ne0 as its ethernet interface
pppoectl -e ne0 pppoe0

# Configure authentication
pppoectl pppoe0 myauthproto=pap myauthname=XXXXX \
myauthsecret=YYYYY  hisauthproto=none

# Configure the pppoe0 interface itself. These addresses are magic,
# meaning we don't care about either address and let the remote
# ppp choose them.
ifconfig pppoe0 0.0.0.0 0.0.0.1 netmask 0xffffffff up
```

In order to be configured as a PPPOE server, you should set the `pppoe` interface to use `link0` mode, so that it can wait for incoming PPPOE session requests. For

Copyright © 2014, QNX Software Systems Limited

example, the following makes the interface wait for requests coming from the `en0` interface:

```
ifconfig pppoe0 create # create an PPPOE interface
ifconfig pppoe0 link0  # set mode as link0
pppoectl -e en0 pppoe0 # link it with en0 ethernet interface
pppoectl  pppoe0 hisauthproto=pap hisauthname=XXX hisauthsecret=YYY
                       # configure authentication
ifconfig pppoe0 inet ip_address_self ip_address_client
                       # assign ip addresses
ifconfig en0 up        # up en0 to receive incoming data.
```

Files:

/dev/io-net/ppp_en

The default PPPOE device.

/etc/ppp/pppoe-down

The default downscript.

/etc/ppp/pppoe-up

The default upscript.

Copyright © 2014, QNX Software Systems Limited

pps

Persistent Publish/Subscribe manager (QNX Neutrino)

Syntax:

```
pps [options]
```

Runs on:

QNX Neutrino

Options:

-b

Do *not* run in the background. Useful for debugging.

-l *argument*

Set the object load behavior, as follows:

- 0 — load directory names and objects on demand. Default.
- 1 — load all directory and object names on startup, but do not load object contents. Load object contents on demand.
- 2 — load directories, objects, and object contents on startup.

-m *mount*

Specify the mountpath for PPS. Default is `/pps/`

-p *path*

Set the path for backing up the persistent storage.

-t *period*

Specify the periodicity of the forced persistence, in milliseconds. For example `-t 5000` forces the PPS service to write to persistent storage every five seconds. Default is no forced persistence.

-v

Enable verbose mode. Increase the number of "v"s to increase verbosity.

Copyright © 2014, QNX Software Systems Limited

Description:

The pps program manages the Persistent Publish/Subscribe system, which provides a simple method of disseminating information to interested processes.

Copyright © 2014, QNX Software Systems Limited

pr

Break a file into pages and/or columns for printing

Syntax:

```
pr [+page] [-column] [-adFmrt] [-e[char][gap]]
[-L locale] [-h header]
[-i[char][gap]] [-l lines]
[-o offset] [-s[char]] [-n[char][width]]
[-w width] [file …]
```

Runs on:

QNX Neutrino

Options:

In the following option descriptions, *column*, *lines*, *offset*, *page*, and *width* are positive decimal integers, and *gap* is a nonnegative decimal integer.

Options -e, -i, -n, and -s, do not use spaces to separate options and arguments.

+page

Print the pages starting with this *page* number.

-column

Produce output that is *columns* wide (the default is 1). The text is written vertically down each column in the order in which it is received from the input file. The options -e and -i are assumed. Don't use this option with -m. If you want to display the output using the minimum number of lines, use with -t.

-a

Modify the effect of the -*column* option so that the columns are filled across the page in a round-robin order (e.g. when *column* is 2, the first input line heads column 1, the second heads column 2, the third is the second line in column 1, etc.). If you use this option, you must also use the -*column* option.

-d

Copyright © 2014, QNX Software Systems Limited

Double space the output. An extra newline character is output following every newline found in the input.

-e[*char*][*gap*]

Expand each input tab to the next greater column position specified by the formula n**gap*+1, where n is an integer > 0. If *gap* is zero or is omitted, the default is 8. All tab characters in the input are expanded into the appropriate number of spaces. If *char* (any nondigit character) is specified, it is used as the input tab character.

-F

Use a form-feed character for new pages, instead of the default behavior that uses a sequence of newline characters.

-h *header*

Use the string *header* to replace the file name in the header line.

-i[*char*][*gap*]

In the output, replace multiple spaces with tabs whenever two or more adjacent spaces reach column positions *gap*+1, 2**gap*+1, etc. If *gap* is zero or omitted, the default tab settings are at every eighth column position. If *char* (any nondigit character) is specified, it is used as the output tab character.

-L *locale*

Use the *locale* specified as argument instead of one found in the environment. Use C to reset *locale* to its default.

-l *lines*

("el") Override the 66-line default and reset the page length to *lines*. If *lines* is not greater than the sum of both the header and trailer depths (in lines), the pr utility suppresses output of both the header and trailer, as if the -t option were in effect.

-m

Merge the contents of multiple files. One line from each file specified by a file operand is written side by side into text columns of equal fixed widths, in terms of the number of column positions. The number of text columns depends on the number of file operands successfully opened. The maximum number of files merged depends on page width and the per-process open file limit. The options -e and -i are assumed.

Copyright © 2014, QNX Software Systems Limited

-n[*char*][*width*]

Provide *width*-digit line numbering. The default for *width*, if not specified, is 5. The number occupies the first *width* column positions of each text column or each line of -m output. If *char* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows. The default for *char* is a tab. Line numbers longer than *width* columns are truncated.

-o *offset*

Each line of output is preceded by *offset* spaces. If the -o option is not specified, the default is zero. The space taken is in addition to the output line width.

-r

Inhibit the warning when a file can't be opened.

-s[*char*]

Separate text columns by the single character *char* instead of by the appropriate number of spaces (the default for *char* is the tab character).

-t

Print neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.

-w *width*

Set the width of the line to *width* column positions for multiple text-column output only. If the -w option is not specified and the -s option is not specified, the default width is 72. If the -w option is not specified and the -s option is specified, the default width is 512.

file

A pathname of a file to be printed. If no *file* operands are specified, or if a *file* operand is "-", the standard input is used.

Description:

The pr utility is a printing and pagination filter for text files. When multiple input files are specified, each is read, formatted, and written to standard output. By default, the input is separated into 66-line pages, each with:

- A 5-line header with the page number, date, time, and the pathname of the file.
- A 5-line trailer consisting of blank lines.

Copyright © 2014, QNX Software Systems Limited

If standard output is associated with a terminal, diagnostic messages are suppressed until the `pr` utility has completed processing.

When multiple column output is specified, text columns are of equal width. By default text columns are separated by at least one space. Input lines that do not fit into a text column are truncated. Lines are not truncated under single column output.

Exit status:

`0`

Success.

`1`

An error occurred.

Errors:

If `pr` receives an interrupt while printing to a terminal, it flushes all accumulated error messages to the screen before terminating. Error messages are written to *stderr* during the printing process (if output is redirected) or after all successful file printing is complete (when printing to a terminal).

Contributing author:

Keith Muller

License:

This utility is based on copyright software of The Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Copyright © 2014, QNX Software Systems Limited

/etc/printcap

Printer-capability database

Name:

/etc/printcap

Description:

The printcap database is a simplified version of the termcap database for describing printers. The spooling system accesses the printcap file every time it's used, so you can dynamically add or remove printers. Each entry in the database describes one printer.

Note that you can't replace this database — as you can for termcap — because that might allow accounting to be bypassed.

The default printer is normally lp, although you can use the environment variable ***PRINTER*** to override this. Each spooling utility supports a -P*printer* option to explicitly name a destination printer.

Each entry in the printcap file describes a printer. An entry is a line consisting of a number of fields separated by : characters. The first entry for each printer gives the known names for the printer, separated by | characters.

The first name is conventionally a number. The second name is the most common abbreviation for the printer; the last name should be a long name fully identifying the printer. The second name should contain no blanks; the last name may well contain blanks for readability. Entries may continue onto multiple lines by giving a \ as the last character of a line. Empty fields may be included for readability.

Capabilities in printcap are all introduced by two-character codes and are of these types:

Boolean

Indicates that the printer has some particular feature. Boolean capabilities are simply written between the : characters.

Numeric

Information such as baud rate, number of lines per page, and so on. Numeric capabilities are given by the two-character capability code followed by the # character, followed by the numeric value. The following example is a numeric entry stating that this printer should run at 1200 baud:

```
:br#1200:
```

Copyright © 2014, QNX Software Systems Limited

String

A sequence that can be used to perform particular printer operations such as cursor motion. String-valued capabilities are given by the two-character capability code followed by an = sign and then a string ending at the next following `:`. For example:

```
:rp=spinwriter:
```

states that the remote printer is named `spinwriter`.

Capabilities

Name	Description	Type	Default
`af`	Name of accounting file	String	`NULL`
`br`	If lp is a tty, set the baud rate (*ioctl()* call)	Numeric	None
`cf`	`cifplot` data filter	String	`NULL`
`df`	TeX data filter (DVI format)	String	`NULL`
`du`	Userid of user `daemon`	String	0
`fc`	If lp is a tty, clear flag bits	Numeric	0
`ff`	String to send for a form feed	String	`\f`
`fo`	Print a form feed when device is opened	Boolean	False
`fs`	Similar to `fc` but set bits	Numeric	0
`gf`	Graph data filter (plot(3X) format)	String	`NULL`
`hl`	Print the burst header page last	Boolean	False

Copyright © 2014, QNX Software Systems Limited

Name	Description	Type	Default
ic	Driver supports (non standard) *ioctl()* to indent printout	Boolean	False
if	Name of input/communication filter (created per job)	String	NULL
lf	Error logging filename	String	/dev/console
lo	Name of lock file	String	lock
lp	Device name to open for output	String	/dev/lp
mc	Maximum number of copies	Numeric	0
ms	List of terminal modes to set or clear	String	NULL
mx	Maximum file size (in BUFSIZ blocks); zero = unlimited	Numeric	1000
nf	ditroff data filter (device independent troff)	String	NULL
of	Name of output/banner filter (created once)	String	NULL
pc	Price per foot or page in hundredths of cents	Numeric	200
pl	Page length (in lines)	Numeric	66
pw	Page width (in characters)	Numeric	132
px	Page width in pixels (horizontal)	Numeric	0

Copyright © 2014, QNX Software Systems Limited

Name	Description	Type	Default
py	Page length in pixels (vertical)	Numeric	0
rf	Filter for printing FORTRAN style text files	String	NULL
rg	Restricted group — only members of group allowed access	String	NULL
rm	Machine name for remote printer	String	NULL
rp	Remote printer name argument	String	lp
rs	Restrict remote users to those with local accounts	Boolean	False
rw	Open printer device read/write instead of write-only	Boolean	False
sb	Short banner (one line only)	Boolean	False
sc	Suppress multiple copies	Boolean	False
sd	Spool directory	String	/usr/spool/output/lpd
sf	Suppress form feeds	Boolean	False
sh	Suppress printing of burst page header	Boolean	False
st	Status filename	String	status
tc	Name of similar printer; must be last	String	NULL
tf	troff data filter (C/A/T phototypesetter)	String	NULL

Copyright © 2014, QNX Software Systems Limited

Name	Description	Type	Default
tr	Trailer string to print when queue empties	String	NULL
vf	Raster image filter	String	NULL
xc	If lp is a tty, clear local mode bits	Numeric	0
xs	Like xc but set bits	Numeric	0

If the local line printer driver supports indentation, the daemon must understand how to invoke it.

Note that the fs, fc, xs, and xc fields are flag *masks* rather than *values*. Certain default device flags are set when the device is opened by the line printer daemon if the device is connected to a terminal port. The flags indicated in the fc field are then cleared; the flags in the fs field are then set (or vice versa, depending on the order of fc#*nnnn* and fs#*nnnn* in the /etc/printcap file).

The bits cleared by the fc field and set by the fs field are those in the sg_flags field of the sgtty structure, as set by the TIOCSETP *ioctl()* call, and the bits cleared by the xc field and set by the xs field are those in the "local flags" word, as set by the TIOCLSET *ioctl()* call. See <sys/termio.h> for a description of these flags.

For example, to set exactly the flags 06300 in the fs field, which specifies that the EVENP, ODDP, and XTABS modes are to be set and that all other flags are to be cleared, do:

```
:fc#0177777:fs#06300:
```

The same process applies to the xc and xs fields. Alternatively, you can use the ms field to specify modes to be set and cleared. These modes are specified as *stty* (p. 1863) modes; you can specify any mode supported by stty, except for the baud rate (which you must specify with the br field).

For example, to set the terminal port (to which the printer is attached) to even parity, TAB expansion, no NEWLINE to RETURN/LINEFEED translation, and RTS/CTS flow control enabled, do the following:

```
:ms=evenp,-tabs,nl,crtscts:
```

The tc field must appear last in the list of capabilities. Each type of printer should have a general entry describing common capabilities. Then an individual printer can be defined with its particular capabilities, plus a tc field that points to the general entry for that type of printer.

Copyright © 2014, QNX Software Systems Limited

For information about setting up the `printcap` file, see the Printing chapter of the QNX Neutrino *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

printf

Write formatted output (POSIX)

Syntax:

```
printf format [argument...]
```

Runs on:

QNX Neutrino

Options:

format

A string describing how the given *arguments* are to be formatted.

argument

A string to be written to standard output, under control of *format*.

Description:

The `printf` utility writes formatted arguments to standard output under control of the *format* control string (the syntax for *format* is based upon the *printf()* function).

The Korn shell has a builtin *print* (p. 1065) command that you can also use to write arguments to standard output. For more information, see the entry for `ksh`.

The *format* control string contains the following types of characters:

- Ordinary characters that are written exactly as they occur in the *format* string
- Escape sequences that represent nongraphic characters
- Conversion specifications that specify the output format of the named *arguments*.

Introduce a conversion specification by the % character. To literally display this character, encode it as %%. Immediately following the % character, you may specify any of the following, in order:

<flags><width><precision><type len><conversion>

where:

flags

Format control flags.

width

Copyright © 2014, QNX Software Systems Limited

Field width.

precision

Numerical precision.

type len

Type length modifier.

conversion

Conversion character.

These components are processed from left to right.

Format control flags

The format control flags modify the meaning of the conversion specification. You can specify zero or more of these flags, in any order:

-

Left justify output within the field.

+

Always begin the result of a signed conversion with a sign (+ or -).

Space

If the first character of a conversion *isn't* a sign, prefix a space to the result.

#

Convert the value to an alternate form:

For this conversion:	The result is formatted as follows:
o (octal)	The first digit is zero
X or x (hex)	A nonzero result has `0X` or `0x` prefixed to it
e, E, f, g, and G	The result always has a decimal point (`.`), even if no digits follow
g and G	Trailing zeros aren't removed from the result as they usually are
c, d, i, s, and u	No effect; the # is ignored
0	For all conversions, pad the field width with leading zeros. The 0 flag

Copyright © 2014, QNX Software Systems Limited

For this conversion:	The result is formatted as follows:
	is ignored for the d, i, o, u, X, and x conversions if a precision is specified; it's also ignored if the - flag is given.

Field width

An optional decimal integer that specifies the minimum number of characters for displaying the formatted item. If no field width is specified, or if the value is less than the number of characters required to represent the converted value (subject to the specified precision), a field of sufficient width to contain the converted value is used.

If the number of characters required to represent the converted value is less than the field width, the value is padded on the left (or on the right if the - format control flag is given) with spaces or "0" (zero) characters. If the field width begins with a zero, the value is padded with zeros, otherwise spaces.

Precision

An optional decimal integer following a period character (`.`). The effect of the given precision depends on the conversion character you specify (see "Conversion character," below).

Type length

An optional character that modifies the interpreted size of the conversion character. The only supported type length character is l ("el"), which causes a d, i, o, u, X, or x conversion to process a `long int` or `unsigned long int` argument.

Conversion character

A character that determines the type of conversion to be applied. For example, if d is specified, `printf` treats the corresponding argument as a decimal integer. The conversion characters are:

d i o u X x

The argument is an integral value. It's printed as one of the following:

- signed decimal (d or i)
- unsigned octal (o)
- unsigned decimal (u)
- unsigned hexadecimal (X or x)

For each of these, the specified precision is interpreted as the minimum number of digits to appear (defaults to 1 if no precision is specified). In the absence of a type length specifier, these conversions are either signed in

Copyright © 2014, QNX Software Systems Limited

the range -32768 to 32767 or unsigned in the range 0 to 65535. If a type length is specified, the conversions are either signed in the range -2147483648 to 2147483647 or unsigned in the range 0 to 4294967295. For the hex format, the case of the specifier (X or x) implies the case of the output.

e, E

The argument is a floating-point value, and is displayed in scientific notation. The value is printed in the style:

```
[-]d[.ddd]e{+|-}dd
```

The first part is an optional sign. The second is a single digit, which is nonzero if the argument is nonzero. The third (optional) part — a decimal character followed by a number of digits equal to the specified precision — is printed if the precision is nonzero. If the precision isn't specified, it's assumed to be 6. If the precision is 0, the decimal isn't printed. The fourth part is an e or E, based upon the case of the conversion character, followed by a signed value. This value represents the order of magnitude.

f

The argument is a floating-point value. The value is printed in the style:

```
[-]ddd.ddd
```

where the number of digits printed after the decimal character is equal to the specified precision. If the precision isn't specified, it's assumed to be 6. If the precision is 0, the decimal isn't printed.

g, G

The argument is a floating-point value. The value is printed in the “e” style (or “E” style if a G conversion character is given) if the exponent in this style is less than -4 or greater than or equal to the specified precision; otherwise, the value is printed in the “f” style.

c

The first character of the argument is printed.

s

The argument is a string of characters and the string is printed. The number of characters printed from the string is equal to the precision, if specified. The string is left- or right-justified within the specified field width. If no field width is specified, no padding is applied.

Copyright © 2014, QNX Software Systems Limited

Escape sequences

The `printf` utility interprets the character escape sequences within the *format* string in a way similar to the C programming language. These sequences, which you introduce via the backslash character (`\`), are translated as follows:

This sequence:	Writes:
`\a`	An alert character (bell)
`\b`	A backspace character
`\f`	A form-feed character
`\n`	A newline character
`\r`	A carriage return character
`\t`	A tab character
`\v`	A vertical tab character
`\'`	A single quote (`'`) character
`\\`	A backslash (`\`) character
`\`*d*	The character specified by the one-, two-, or three-digit octal number *d*

Examples:

Display the string "`hello, world`" on a line by itself:

```
printf "hello, world\n"
```

Do the same as above, but don't output a newline:

```
printf "hello, world"
```

Display a value in scientific notation:

```
printf "n = %e\n" 3.1415926535897
```

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

procnto*

microkernel and process manager (QNX Neutrino)

Syntax:

```
procnto* [-a d|e|s] [-c] [-e n|o] [-F number] [-fe]
         [-h] [-H size] [-mmemmgr_configuration]
         [-P priority[s]] [-p] [-T timeout] [-u umask] [-v]
```

Runs on:

QNX Neutrino

Options:

-ad

Disable alignment fault emulation. The `procnto` manager doesn't attempt to make misaligned memory accesses work; they'll cause a `SIGBUS` signal for the offending thread.

-ae

Enable alignment fault emulation. The `procnto` manager attempts to make misaligned memory accesses work, although they'll be slow. This isn't guaranteed to work; offending threads may still get a `SIGBUS` signal.

In QNX Neutrino 6.6 and later on ARMv7 processors, specifying this option allows misaligned accesses to be performed in hardware. On these targets, `ThreadCtl(_NTO_TCTL_ALIGN_FAULT, ...)` no longer changes the alignment-fault behavior on a per-thread basis.

-as

Use the system default for alignment faults. This behavior depends on your platform:

Platform	System default
ARM	-ad
x86	-ae

-c

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino Core OS 6.3.2 or later) Prevent the adaptive partitioning scheduler from automatically running threads that receive events from interrupt handlers as critical.

The -c option has an effect only if:

- you're using Adaptive Partitioning
- the thread receiving an event from an interrupt handler is running in a partition configured with a critical budget

For more information, see the Adaptive Partitioning *User's Guide*.

-e nlo

(QNX Neutrino 6.4.0 or later) Specify which value to use for EALREADY:

- -eo — use the old value, which is the same as that of EBUSY.
- -en — use the POSIX-compliant value.

In QNX Neutrino 6.4.0, the default value for EALREADY was the old one (-eo); in QNX Neutrino 6.6.0, it's changed to the new value (-en).

For more information, see "Changes to EALREADY" in the entry for *errno* in the QNX Neutrino *C Library Reference*.

-F *number*

The maximum number of file descriptors that can be open at the same time. The minimum allowable value is 100. The default value is 1000, but might be constrained by the `RLIMIT_NOFILE` system resource.

Sockets, named semaphores, message queues, channel IDs (chids), and connection IDs (coids) all use file descriptors.

To determine the current limit, use the `ksh` builtin command, *`ulimit`* (p. 1078), or call *getrlimit()* (see the QNX Neutrino *C Library Reference*).

-fe

Force floating-point emulation. The default is to use floating-point hardware in the CPU if present, and to emulate it in software if the CPU has no FPU (see *`fpemu.so`* (p. 783)). Use this option to simulate a system with no FPU.

-h

Copyright © 2014, QNX Software Systems Limited

Disable CPU halting in idle thread. Some CPU and supporting chipsets can lock up if the CPU halts when idle; you'll notice the need for the -h option right away because your system will lock up after booting.

-H *size*

Sets the initial heap size for procnto. If more memory is required for procnto, it's dynamically obtained; however, by setting a properly calculated value this option can speed up boot time, and reduce the amount of physical memory fragmentation.

The *size* parameter indicates the number of bytes to grow the heap in advance. You can postfix this value with a multiplier character, such as “k” (kilobyte) or “m” (megabyte). For example:

```
1m == 1024k == 0x100000
```

If the number is less than 1024 and it isn't postfixed by a multiplier character, it's assumed to be in kilobytes. The default value is 64 KB if the -H option isn't specified.

-m *memmgr_configuration*

(QNX Neutrino Core OS 6.3.2 or later) Control the behavior of the memory manager. The *memmgr_configuration* string is a sequence of characters that enable (or if preceded with a ~ character, disable) memory-manager aspects.

If you specify more than one -m option, `procnto` ignores all but the last one.

The configuration options are:

a

(QNX Neutrino 6.4.1 or later) Automatically mark memory pages that have a *mem_offset()* performed on it as unmovable when defragmenting physical memory. This has an effect only if defragmenting is enabled (see the -md option below). For more information, see “Automatically marking memory as unmovable” in the Process Manager chapter of the *System Architecture* guide.

~a

(QNX Neutrino 6.4.1 or later) Disable the automatic marking of memory pages as unmovable (the default).

b

Enable backward compatibility (the default).

Copyright © 2014, QNX Software Systems Limited

See the release notes for the current behavior.

~b

Disable backward compatibility.

c

(QNX Neutrino 6.6 or later) Clear memory when it's freed.

~c

(QNX Neutrino 6.6 or later) Don't clear memory when it's freed (the default). When memory is freed for later reuse, the contents of that memory remain untouched; whatever the application that owned the memory left behind is left intact until the next time that memory is allocated by another process. At that point, before the memory is handed to the next process, it's zeroed.

d

(QNX Neutrino 6.4.1 or later) Enable the defragmenting of physical memory (the default). For more information, see "Defragmenting physical memory" in the Process Manager chapter of the *System Architecture* guide.

~d

(QNX Neutrino 6.4.1 or later) Disable the defragmenting of physical memory.

i

Make *munmap()* act as if `UNMAP_INIT_REQUIRED` were specified (i.e., POSIX initialization of the page to all zeroes is required the next time the underlying physical memory is allocated). This is the default.

~i

Make *munmap()* act as if `UNMAP_INIT_OPTIONAL` were specified (i.e., initialization of the underlying physical memory to zeroes on its next allocation is optional). See "Initializing allocated memory" in the Interprocess Communication (IPC) chapter of the *System Architecture* guide.

Copyright © 2014, QNX Software Systems Limited

l ("el")

Lock all memory; act as if `mlockall(MCL_CURRENT|MCL_FUTURE)` were specified at the start of every program. For more information, see *mlockall()* in the in the QNX Neutrino *C Library Reference*.

~l

Don't lock all memory (the default).

L

Superlock all memory; act as if `ThreadCtl(_NTO_TCTL_IO,0)` were specified at the start of every program (but only insofar as locking the memory; programs don't actually get I/O privileges).

~L

Don't superlock all memory (the default).

If you enable both l and L, the L option takes priority.

P

Turn on full allocation of high memory for all processes. This is mostly useful only for testing.

~P

Make sure that all anonymous allocation occurs below the 4 GB mark (the default).

r

(QNX Neutrino 6.5.0 or later; not supported by the QNX Neutrino RTOS Safe Kernel 1.0) Enable address space randomization. If you use this option, the kernel places certain items (e.g., the stack, `libc`) at different addresses every time you run a process. This can help prevent someone from hacking into a program.

~r

(QNX Neutrino 6.5.0 or later; not supported by the QNX Neutrino RTOS Safe Kernel 1.0) Disable address space randomization (the default).

Copyright © 2014, QNX Software Systems Limited

v

Enable variable page sizes (the default). This automatically allows for mapping to be performed with different page sizes to achieve better performance.

~v

Disable variable page sizes.

X

(QNX Neutrino 6.6 or later) Fail any attempts by *mmap()* or *mprotect()* to turn on PROT_EXEC for a memory-mapped file mapping if the file doesn't have execute permission for the client process; an error of EACCES will be given instead.

~X

(QNX Neutrino 6.6 or later) Allow *mmap()* or *mprotect()* to turn on PROT_EXEC for a memory-mapped file mapping even if the file doesn't have execute permission for the client process.

x

(QNX Neutrino 6.4.0 or later) Enable the PROT_EXEC flag for system-allocated threads (the default). This option allows *gcc* (p. 889) to generate code on the stack — which it does when taking the address of a nested function (a GCC extension).

~x

(QNX Neutrino 6.4.0 or later) Turn off PROT_EXEC for system-allocated stacks, which increases security but disallows taking the address of nested functions. You can still do this on a case-by-case basis by doing an *mprotect()* call that turns on PROT_EXEC for the required stacks.

-P *priority*[s]

Set the lower end of the range of privileged priorities to the given *priority*; the upper end of the range is 255. Only processes with an effective user ID of 0 (i.e., root) or those with the PROCMGR_AID_PRIORITY ability enabled (see *procmgr_ability()*) can use these priorities. Unprivileged (and privileged) processes can use priorities from 1 through *priority* – 1. The default value of *priority* is 64; the minimum permitted value is 10, and the maximum is 255.

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino 6.6 or later) You can append an s or S to the *priority* if you want out-of-range priority requests by default to saturate at the maximum allowed value instead of resulting in an error.

-p

Disable kernel preemption. This prevents threads running in kernel space from being preempted by a higher-priority thread. This can be useful when debugging a system with a frequent source of high-priority interrupts.

-T *timeout*

(QNX Neutrino 6.4.0 or later) Specify the number of seconds to wait for a *close()* to succeed in the event of a process termination. Previously, this timeout was hard coded to be 30 seconds. The current default is also 30 seconds; however, the -T *timeout* option lets you set this value.

When a process terminates, any outstanding connections are closed. This means that an *_IO_CLOSE* message is synthesized and sent to the resource manager responsible for that connection.

Because it is not guaranteed that the server will reply in a reasonable amount of time (e.g., a Qnet node may be down), a *TimerTimeout()* call before the send guarantees that the termination process will proceed.

-u *umask*

(QNX Neutrino 6.4.0 or later) Use the given file-creation mask (*umask* (p. 2005)) when creating the entries in `/proc/`*pid*`/as`. If you don't specify this option, `procnto` uses a mask of 0066.

Opening `/proc/`*pid*`/as` for read-only access succeeds, even if the file permissions would normally say that it should fail with an EACESS. Instead the kernel marks the OCB as allowing only *devctl()* commands. This prevents unprivileged processes from examining a process's memory, but still allows a non-`root pidin` to display some useful information.

For more information about these files, see "*/proc filesystem* (p. 1593)," below.

-v[v]...

Be verbose. Specifying more `v` characters increases the verbosity. If you specify this option, you'll get more useful information when a process is terminated by a signal.

Copyright © 2014, QNX Software Systems Limited

Description:

The `procnto` system process contains the microkernel, process management, memory management and pathname management. It's required in all bootable images made using the *`mkifs`* (p. 1241) utility. For more information, see the QNX Neutrino *System Architecture* guide.

To determine the release version of the kernel on your system, use the *`uname -a`* (p. 2010) command.

There are different versions of `procnto` for different processors (see the Board Support Package for your board for specific information):

`procnto-smp`

All other supported multicore processors.

`procnto`

All other supported processors.

There's also an instrumented version of each of the above (e.g., `procnto-smp-instr`) that you'll use for system analysis. For more information, see the System Analysis Toolkit *User's Guide* and the Analyzing Your System with Kernel Tracing chapter of the IDE *User's Guide*.

If you're using an SMP version of `procnto`, you can use the appropriate `startup-*` command's -P option to specify the maximum number of CPUs to activate.

Starting in QNX Neutrino 6.3.0, `procnto` also manages named semaphores, which *`mqueue`* (p. 1319) used to do (`mqueue` now manages named semaphores only if it detects that `procnto` isn't doing so). Named semaphores appear in the pathname space under `/dev/sem`. The *sem_** client functions handle named semaphores; for more information, see the QNX Neutrino *C Library Reference*.

`/proc` filesystem

The Process Manager component of `procnto` implements a `/proc` filesystem that includes the following:

`/proc/`*`pid`*

Virtual directories that let you access and control every process and thread running within the system. For more information, see "Controlling processes via the `/proc` filesystem" in the Processes chapter of the QNX Neutrino *Programmer's Guide*.

`/proc/boot/`

Copyright © 2014, QNX Software Systems Limited

The image filesystem that comprises the boot image. For more information, see Making an OS Image in *Building Embedded Systems*.

/proc/dumper

A special entry that receives notification when a process terminates abnormally. The *dumper* (p. 652) utility watches this entry.

/proc/mount/

Pathname-space mountpoints.

If you list the contents of the `/proc` directory, `/proc/mount` doesn't show up, but you can list the contents of `/proc/mount`.

/proc/qnetstats

If you're using Transparent Distributed Processing (TDP), the *lsm-qnet.so* (p. 1149) module places a `qnetstats` entry in `/proc`. If you open this name and read from it, the Qnet resource manager code responds with the current statistics for Qnet.

/proc/self/

The address space for yourself (i.e., for the process that's making the query).

Examples:

To disable preemption in kernel code:

```
procnto -p
```


Copyright © 2014, QNX Software Systems Limited

/etc/protocols

Protocol name database (UNIX)

Name:

/etc/protocols

Description:

The /etc/protocols file contains information regarding the known protocols used in the DARPA Internet. For each protocol, a single line should be present with the following information:

official_protocol_name protocol_number aliases

Items are separated by any number of blanks or tabs, or both. A # in a line indicates the beginning of a comment — any characters after a #, up to the end of the line, aren't interpreted by routines that search the file.

Protocol names may contain any printable character other than a field delimiter, newline, or comment character.

Copyright © 2014, QNX Software Systems Limited

ps

Report process status (POSIX)

Syntax:

```
ps [-aAdEfl] [-[gG] grplist] [-o format]... [-n namelist]
   [-p proclist] [-t termlist] [-[uU] usrlist]
```

Runs on:

QNX Neutrino

Options:

-a

Write information for all processes associated with terminals.

-A

Write information for all processes.

-d

Write information for all processes, except session leaders.

-e

Write information for all processes (the same as -A).

-f

Generate a full listing.

-G *grplist*

Write information for processes whose real group ID is given in the comma- and space-delimited list, *grplist*.

-g *grplist*

Write information for processes whose session leader is given in the comma- and space-delimited list, *grplist*.

-l

Generate a long listing.

-n *namelist*

Copyright © 2014, QNX Software Systems Limited

The name of an alternate system namelist file. (Not supported)

-o *format*

Write information according to the specifications given in *format*. If you specify more than one -o option, ps concatenates all the *format* arguments.

-p *proclist*

Write information for processes whose ID is given in the comma and space-delimited list, *proclist*.

-t *termlist*

Write information for processes whose terminal identifier is given in the comma and space-delimited list, *termlist*.

-U *usrlist*

Write information for processes whose real user ID or login name is given in the comma and space-delimited list, *usrlist*.

-u *usrlist*

Write information for processes whose user ID or login name is given in the comma and space-delimited list, *usrlist*.

Description:

The ps utility prints information about processes, subject to having the appropriate privilege to obtain information about those processes. With no options, ps prints information about processes associated with the current terminal. The output includes the process ID, terminal name, cumulative execution time, and command name of each process.

The options that use lists (-G, -g, -o, -p, -t, -U, and -u) can list multiple items separated by commas or white space, as long as all the items are contained within a single command-line argument. If white space is used, you'll probably need to quote the list when invoking ps from the shell.

The initial set of processes selected by -a, -A, or -d is intersected with the processes selected by the -G, -g, -p, -t, -U or -u options, if any of the latter are specified. If a process meets any of the selection criteria, its information is displayed.

If none of -a, -A, or -d is specified, ps behaves as though you specified -u *your_uid*.

Controlling output

The -o option controls the information that ps displays. The *format* argument is a comma- or space-delimited list of field names that represent information to be displayed about processes and threads.

Copyright © 2014, QNX Software Systems Limited

You can override the header for a field by appending an equals sign and the new header. The rest of the characters in the argument are used as the header text. The ps utility automatically determines the width of all columns.

The ps utility recognizes the following field names:

ruser

The real user name or real user ID of the process.

user

The effective user name or effective user ID of the process.

rgroup

The real group name or real group ID of the process.

group

The effective user name or effective group ID of the process.

pid

The process ID.

ppid

The parent process ID.

pgid

The process group ID.

pcpu

The ratio of CPU time used to CPU time available in a recent period of time.

vsz

The size of the process.

nice

The effective priority of the process.

etime

The elapsed time since the process was started.

time

The cumulative CPU time of the process.

tty

The name of the controlling terminal. (Currently not supported).

Copyright © 2014, QNX Software Systems Limited

comm

The command being executed.

args

The command with all of its arguments.

The following field names are supported, even though they aren't required by POSIX:

uid

The real user name or real user ID of the process.

env

The environment in a space-delimited list.

f

The process flags.

pri

The real priority of the process.

sid

The process ID of the session leader.

suid

The user ID of the session owner.

sgid

The group ID of the session owner.

umask

The process file creation mask.

sigign

The list of signals ignored by the process.

sigpend

The list of signals pending on the process and threads.

sigqueue

The list of signals queued on the process.

threads

The number of threads for the process.

Copyright © 2014, QNX Software Systems Limited

stime

The process's start time.

cmd

The command being executed.

sz

The size of memory used.

Some of the above fields are thread-specific. In such cases, the value for the first thread in the process is used.

The following field names display information about threads:

tid

The thread ID.

tflags

The thread flags.

dflags

The thread debug flags.

tsigblk

The list of signals blocked on the thread.

cpu

The last CPU the thread ran on.

state

The state of the thread.

wchan

The wait channel.

addr

The wait address.

psched

The process scheduling policy.

The `comm`, `args`, and `env` fields are the only ones that can contain blanks.

If you don't specify a format, `ps` uses:

Copyright © 2014, QNX Software Systems Limited

- `uid,pid,ppid,c,stime,tty,time,args` if you specify -f
- `f,s,uid,pid,ppid,c,pri,nice,addr,sz,wchan,tty,time,cmd` if you specify -l
- `pid,tty,time,cmd` if you don't specify -f or -l

Examples:

List the process ID, cumulative CPU time, and the command with arguments for all processes, overriding the title of the last column:

```
ps -A -o "pid,time,args=Command (with args)"
```

Exit status:

0

Successful completion.

-1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

pwd

Print working directory name (POSIX)

Syntax:

pwd

Runs on:

QNX Neutrino, Microsoft Windows

Options:

None.

Description:

The pwd utility writes the pathname of the current working directory to the standard output.

The pwd command is available both as a shell alias (equivalent to *print "$PWD"* (p. 1065)), and as a standalone utility. For information about the builtin *cd* (p. 1061) and *pwd* (p. 1066) commands, see *ksh* (p. 1029). To make sure you use the executable, specify the full path.

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

python

Object-oriented programming and scripting language

Syntax:

```
python [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Runs on:

Linux, Microsoft Windows

Options:

-c *cmd*

Pass the *cmd* string as the program to run. This option terminates the option list.

-d

Display debugging output from the parser. You can also enable this by setting ***PYTHONDEBUG*** to a nonzero value.

-E

Ignore Python's environment variables (such as ***PYTHONPATH***).

-h

Print a help message, and then exit.

-i

Inspect interactively after running the script, and force prompts, even if *stdin* doesn't appear to be a terminal. You can also enable this by setting ***PYTHONINSPECT*** to a nonzero value.

-m *mod*

Run the specified library module as a script. This option terminates the option list.

-O

Optimize the generated byte code (a tad). You can also enable this by setting ***PYTHONOPTIMIZE*** to a nonzero value.

-OO

Copyright © 2014, QNX Software Systems Limited

Remove doc-strings in addition to performing the -O optimizations.

-Q *arg*

Control the behavior of the division (`/`) operator, where *arg* is one of the following:

- old — use floor division for integral arguments, and true division for floating-point or complex arguments.
- warn — generate a warning if `/` (instead of `//`) is used for integer division.
- warnall — generate a warning whenever `/` is used.
- new — always perform true division.

The default is old.

This option will be removed in version 3.0 of Python, and the `/` operator will always perform true division. For more information, see *http://www.python.org/dev/peps/pep-0238/*.

-S

Don't imply `import site` on initialization.

-t

Issue warnings about inconsistent tab usage.

-tt

Issue errors about inconsistent tab usage.

-u

Use unbuffered binary for *stdout* and *stderr*. See the Python documentation for details on internal buffering related to this option. You can also enable this by setting ***PYTHONUNBUFFERED*** to a nonzero value.

-v

Be verbose (trace import statements). You can also enable this by setting ***PYTHONVERBOSE*** to a nonzero value.

-V

Print the Python version number, and then exit.

-W *arg*

Warning control; *arg* is *action:message:category:module:lineno*.

Copyright © 2014, QNX Software Systems Limited

-x

Skip the first line of source, allowing the use of non-Unix forms of `#!cmd`.

file

Read the program from this script file.

-

Read the program from *stdin* (the default; Python uses interactive mode if run on a tty).

***arg* ...**

Arguments to pass to the program in *sys.argv[1:]*.

Description:

Python is a high-level, object-oriented programming language that supports powerful programming constructs, can be integrated with other languages such as C and C++, and is extendible through the addition of libraries and modules. You can use Python for programs and sophisticated scripts.

Environment variables:

PYTHONCASEOK

Ignore case in `import` statements (Windows).

PYTHONDEBUG

Display debugging output from the parser.

PYTHONHOME

An alternate *prefix* directory (or *prefix*:*exec_prefix*). The default module search path uses *prefix*/`python`*X.X*.

PYTHONINSPECT

Inspect interactively after running the script, and force prompts, even if *stdin* doesn't appear to be a terminal.

PYTHONOPTIMIZE

Optimize the generated byte code (a tad).

PYTHONPATH

A colon-separated list of directories prefixed to the default module search path. The result is stored in *sys.path*.

Copyright © 2014, QNX Software Systems Limited

PYTHONSTARTUP

The file to execute on interactive startup (no default).

PYTHONUNBUFFERED

Use unbuffered binary for *stdout* and *stderr*. See the Python documentation for details on internal buffering.

PYTHONVERBOSE

Be verbose (trace import statements).

Copyright © 2014, QNX Software Systems Limited

Chapter 18
Q

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`qdb`	QNX Database *Developer's Guide*
`qdbc`	QNX Database *Developer's Guide*

The following have been deprecated:

Instead of:	Use:
`qde`	*`run-qde`* (p. 1723)
`QWinCfg`	not applicable; *`run-qde`* (p. 1723) sets up the development environment before starting the IDE

This chapter describes the utilities, etc. whose names start with “Q”.

Copyright © 2014, QNX Software Systems Limited

QCC, qcc

Compile command (QNX)

Syntax:

For C:

```
qcc [options] [operands]
```

For C++:

```
QCC [options] [operands]
```

Runs on:

Linux, Microsoft Windows

Options:

-A *library*[.a]

Build a new archive.

-a *library*[.a]

Add to the given archive.

-ansi

Compile strict ANSI code.

-Bstatic

Link statically against any subsequent libraries on the command line.

-Bdynamic

Link dynamically against any subsequent libraries on the command line.

-C

Preprocessor leaves comments.

-c

Compile only.

-D *name*[=*value*]

Define the symbol *name*, optionally setting its value.

-E

Copyright © 2014, QNX Software Systems Limited

Preprocess to *stdout*.

-EB or -EL

Compile for big or little endian.

-g

Compile with debug.

-I *path*[:*path* ...]

Set the search path for `#include` directives.

-L *path*[:*path* ...]

Set the library search path.

The development tools have been designed to work out of their processor directories (`x86`, etc.). This means you can use the same tool set for any target platform.

If you have development libraries for a certain platform, put them into the platform-specific library directory (e.g. `/x86/lib`), which is where the compiler tools look by default. Don't put the libraries in `/lib` or `/usr/lib`, which are ignored. Alternatively, use the -L option to specify the libraries' location.

-l *library*

("el") Add *library* to the list of libraries to link against. Omit the `lib` prefix and any extension from the library's name. For example, to link against `libsocket`, specify -l socket.

You can specify more than one -l option. The `qcc` configuration files might specify some libraries for you; for example, `qcc` usually links against `libc`.

The application binary interfaces (ABIs) for the Dinkum C++ and GNU C++ libraries are incompatible. If you link a program against both libraries, you'll get a segmentation fault.

-lang-c

Treat as C (the default for `qcc`).

-lang-c++

Treat as C++ (the default for `QCC`).

Copyright © 2014, QNX Software Systems Limited

You need to link C++ programs with the -lang-c++ option in order for exceptions to work.

-M

Generate a mapfile called *output_file*`.map`.

-N *stacksize*[K]

Specify the stack size, in bytes or kilobytes.

-n

Don't execute.

-nopipe

Use temporary files rather than pipes between phases.

-nostartup

Don't use `ld_startup_*` sections.

-nostdinc

Don't include the standard C include paths.

-nostdinc++

Don't include the standard C++ include paths.

-nostdlib

Don't use the `ld_stdlib` section.

-nostdlib++

Don't use the `ld_stdlib++` section.

-O

Do compile-phase optimization.

-o *outfile*

Specify the name of the output file. The default is `a.out`.

Note that the `make` utility, when used with the default settings, produces an output file with the same name as the input file. For example:

```
make file1
```

results in the executable output file `file1`.

Copyright © 2014, QNX Software Systems Limited

-P

Preprocess to *file*`.i` (C) or *file*`.ii` (C++).

-p

Compile with profiling; see "*Profiling* (p. 1615)", below.

-S

Compile, leave assembly in *file*`.s`.

-save-temps

Save intermediate files created during compilation.

-set-default

Set the current -V argument as the default target. For example:

```
qcc -V4.7.1,gcc_ntoarmv7le -set-default
```

will set GCC 4.7.1 for ARMv7 little-endian as the default.

You must be `root` to use the -set-default option. Note also that the option is intended for use at the command line, not for makefiles.

-shared

When compiling, make the object position-independent so that it's suitable for inclusion in a shared object. When linking, combine the modules into a shared object.

-static

Link against static libraries only.

-std=*language*

Specify the language standard. For more information, see *http://gcc.gnu.org/onlinedocs/gcc-4.7.3/gcc/C-Dialect-Options.html* in the GNU documentation.

-U *name*

Undefine the given symbol.

-V [[*compiler/*]*version*,][*target*]

The compiler name, version number, and the target name. If you don't specify -V at all, `qcc` will use the default compiler, version, and target.

Copyright © 2014, QNX Software Systems Limited

If you specify the *target*, qcc looks for the target configuration files in the following paths, according to how *compiler*, *version*, and *target* are specified:

If you specify:	qcc looks here:
-V*target*	${***QCC_CONF_PATH***}/*compiler*/*version* (where *compiler* is inferred from *target*, and *version* is the default).
-V*version*,*target*	${***QCC_CONF_PATH***}/*compiler*/*version* (where *compiler* is inferred from *target*; if that path doesn't exist, or if *version* contains a slash [/], then ${***QCC_CONF_PATH***}/*version* is used).
-V*compiler*,*target*	${***QCC_CONF_PATH***}/*compiler*/*version* (where *version* is the specified compiler's default version).
-V*compiler*/*version*,*target*	${***QCC_CONF_PATH***}/*compiler*/*version*

For example, this command:

```
qcc -Vgcc,
```

lists all targets in all versions of gcc. See the *Examples* (p. 1616) section below for more examples.

The targets include:

- gcc_ntoarmv7le
- gcc_ntox86

For a list of supported targets, specify -V (or -V*compiler* or -V*compiler*/*version* or -V*version*, provided there's a valid ${***QCC_CONF_PATH***}/*version* directory).

-v[v]

Operate verbosely (the second v turns on verbosity in the compiler).

-W *phase*,*arg*[,*arg* ...]

Pass the specified option and its arguments through to a specific phase:

- p — preprocessor
- c — compiler
- l — linker
- a — assembler.

Copyright © 2014, QNX Software Systems Limited

For example, if you want to pass the -MD option to the compiler, specify -Wc,-MD on the command line for `qcc`.

-w

Suppress all warnings (same as -w0).

If you specify -w along with multiple warning options, *all* warnings are suppressed regardless of the order of the options. In other words, -w always wins.

-w[0-9]

Set the warning level (0=off). The -w9 option is the same as `gcc`'s -W9 option. To generate warnings for *everything*, include these options with -w9:

- -Wcast-qual
- -Wpointer-arith
- -Wshadow
- -Wwrite-strings

Note that using these options will produce warnings in the standard C++ library headers, and possibly other system headers.

-x *extension*

Treat the files that follow as being of type *extension*. The following values of *extension* are accepted:

- `c`
- `c-header`
- `c++`
- `c++-header`
- `cpp-output`
- `assembler`
- `assembler-with-cpp`

as well as valid file extensions, such as `.c`, `.cc`, `.cpp`, `.C`, `.i`, `.ii`, `.s`, and `.S`.

Use -xnone to go back to normal suffix typing. For example, to compile `myfile.h` as if it were a `.c` file:

```
qcc -xc myfile.h
```

-Y *lib_type*

Copyright © 2014, QNX Software Systems Limited

Select the C++ library type to be used (if available), *lib_type* can be:

- `_gpp` — GNU C++ library
- `_cpp` — Dinkum C++ library (default)
- `_cpp-ne` — Dinkum C++ library (no exceptions)

Even with exceptions disabled, the *new()* operator throws a `std::out_of_memory` exception if there isn't enough memory. If you want *new()* to return `NULL` instead of throwing an exception, overload the *new()* operator with your own.

Description:

`QCC` and `qcc` are the QNX compiler interface. They're based on the POSIX `c89` utility. By default, `QCC` considers a program to be C++, while `qcc` considers it to be C.

`QCC` and `qcc` take a list of source and object modules on the command line and invoke the appropriate parser to compile each file. Object modules are passed straight through to the linker. The file suffix determines which parser is used, as follows:

Suffix	Parser
`.s`	Assembler
`.S`	Assembler with preprocessor directives
`.c`	C compiler
`.i`	Preprocessed C file
`.C`, `.cc`, `.cpp`	C++ compiler
`.H`, `.HPP`, `.h++`, `.h`, `.hh`, `.hpp`, `.hxx`	Precompiled header
`.ii`	Preprocessed C++ file
`.o`	Object file
`.a`	Library file

These utilities don't allow multiple options to be specified after a dash character (-). For example, -gc isn't valid; you must specify -g -c instead. Operands (source and object files) and options may be mixed and specified in any order. Some options, such as -I and -L, are order-dependent—the order in which they appear in the command line determines the order of the searches made. All command-line arguments are processed before any compilation or linking begins.

Copyright © 2014, QNX Software Systems Limited

The single-pass linker resolves symbols from left to right: If a module refers to a symbol that is contained in a library, make sure you specify the library to the right of the module.

When `qcc` encounters a compilation error that stops an object file from being created, it writes a diagnostic to the standard error and continues to compile other source files, but it bypasses the link phase and returns a nonzero exit status. If the link phase is entered and is unsuccessful, a diagnostic is written and `qcc` exits with a nonzero status.

The -c option suppresses the link phase. If you have many separate source files that you must update and modify individually, you'll probably use the -c option frequently.

You may occasionally wish to examine the assembly code produced by the code generator. The -S option produces an assembly file ending in `.s`.

If you need to specify a parameter to any of the language processors, you may use the -W *c,option*. Check the documentation on each processor to determine its options.

The compiler defines various preprocessor symbols (or manifest constants) that can help you make decisions at compile time. For more information, see the Manifests chapter of the QNX Neutrino *C Library Reference*.

Profiling

Here's how to profile your application, so you can see where it's spending its time:

1. Compile and link your application with profiling by using the -p option to `qcc`. For example:

   ```
   make CCOPTS+=-p LDOPTS+=-p
   ```

2. Slay *qconn* (p. 1621), or it will redirect the output.
3. Run your application as `root` (this is important because the timers are privileged). The result of this run is a file called `gmon.out`, in your program's current working directory *when it exits*.
4. Look at the profiled output with the command:

   ```
   gprof [your_app] | less
   ```

For more information, see the GNU documentation for `gprof`, and Profiling an Application in the IDE *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

Examples:

Compile `myfile.c` and create a 32-bit executable program for QNX Neutrino on an Intel x86 machine in the current directory with the name `a.out`:

```
qcc -Vgcc_ntox86 myfile.c
```

Compile `myfile.c` and create a 32-bit executable program for QNX Neutrino on an Intel x86 machine in the current directory with the name `myfile`:

```
qcc -Vgcc_ntox86 -o myfile myfile.c
```

Use the default compiler, version, and target:

```
qcc
```

The default compiler is `gcc`, the default version of the compiler is 4.2, and the default target is `ntox86`.

Use the default version of the compiler, and build for an ARM little-endian target:

```
qcc -Vgcc_ntoarmv7le
```

Use the 4.7.1 version of the compiler, and build for an ARMv7 little-endian target:

```
qcc -V4.7.1,gcc_ntoarmv7le
```

Use `make` to compile `myfile.c` and create an executable program in the current directory with the name `myfile`:

```
make myfile
```

You can't use the default rules for `make` — you need to specify the target. See *make* (p. 1166).

Make a shared library:

```
qcc -Vgcc_ntox86 -shared -c shared.c
qcc -Vgcc_ntox86 -shared -o libshared.so shared.o
```

Files:

`a.out`

The default output file. You can use the -o option to override this.

Configuration files:

${***QCC_CONF_PATH***}/*version*/`*.conf`, ${***QCC_CONF_PATH***}`/default`, ${***QCC_CONF_PATH***}`/gcc/default`, ${***QCC_CONF_PATH***}`/gcc/`*version*`/default`

Environment variables:

Copyright © 2014, QNX Software Systems Limited

QCC_CONF_PATH

The name of the directory that contains the configuration files. The default directory is `${QNX_HOST}/etc/qcc`.

Exit status:

0

Success.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

qconfig

Query and display QNX installations and configurations

Syntax:

```
qconfig [-abc] [-d path] [-e | -i | -l] [-h]
        [-n installation_name] [-p] [-r program]
```

Runs on:

Linux, Microsoft Windows

Options:

-a

Display all installed products and updates in a machine-readable format.

The -a option was added in QNX Neutrino Core OS 6.3.2 and an update to QNX Momentics 6.3.0 SP1 and QNX Momentics 6.3.0 SP2.

-b

Display the installed product baseline.

The -b option was added in QNX Neutrino Core OS 6.3.2 and an update to QNX Momentics 6.3.0 SP1 and QNX Momentics 6.3.0 SP2.

-c

Print the runtime environment strings in `csh` style.

The -c option was added in QNX Momentics 6.3.0 SP3 and an update to QNX Momentics 6.3.0 SP1 and QNX Momentics 6.3.0 SP2.

-d *path*

The name of the directory where all of the individual configuration files are located. There's one configuration file per installation.

-e

Copyright © 2014, QNX Software Systems Limited

Set up the runtime environment (see the examples). If you specify this option, `qconfig` produces the commands to set these environment variables:

- ***QNX_HOST***
- ***QNX_TARGET***
- ***PATH***
- ***LD_LIBRARY_PATH***
- ***MAKEFLAGS***

You'll find this option useful when setting up the environment for building software.

-i

List all installations in an easy-to-parse form. The form is:

```
count: number_of_installations
name: ...
version: ...
host: ...
target: ...
```

There's a set of lines for each version of software that you've installed.

-l

List all installations in human-readable format (the default).

-n *installation_name*

Select a specific installation by name. The name may be incorporated as part of the configuration file using the <name> tag or, if not present, it defaults to the filename.

-p

Display the installed product updates.

The -p option was added in QNX Neutrino Core OS 6.3.2 and an update to QNX Momentics 6.3.0 SP1 and QNX Momentics 6.3.0 SP2.

-r *program*

Run the given program in the environment.

Copyright © 2014, QNX Software Systems Limited

Description:

The `qconfig` utility sets or queries the QNX configuration files. It supports the coexistence of multiple versions of QNX Neutrino on one machine.

Whenever you install a version of QNX Neutrino, a configuration file is stored in a directory that depends on the host OS. Each configuration file defines the name of the installation and its base, host, and target directories.

The ***QNX_CONFIGURATION*** environment variable usually identifies the directory where the configuration files are stored, but you can override it by specifying the -d option. If neither of these produces a valid location, `qconfig` looks in `/etc/qnx/qconfig` before giving up and returning an error.

If you don't specify an installation with the -n option, `qconfig` uses the most current installation.

You can use `qconfig` to query your current configuration in a human-readable format, or you can use the -e option to set up your environment to use a certain installation.

This utility doesn't list the installed packages in any particular order.

Examples:

List all the QNX installations:

```
qconfig
```

Set up your shell environment for a specific installation:

```
eval `qconfig -n "QNX Software Development Platform 6.6.0" -e`
```

The syntax for evaluating the output of the -e option depends on the host OS and the shell that you're using. The command shown above works with *ksh* (p. 1029) and `bash`.

Run a specific instance of `qde` (the IDE):

```
qconfig -n "QNX Software Development Platform 6.6.0" -r qde
```

Environment variables:

QNX_CONFIGURATION

The name of the directory that stores the configuration files.

Copyright © 2014, QNX Software Systems Limited

qconn

Provide service support to remote IDE components

Syntax:

```
qconn [-v] [port=portnum]
      [qconn_prio=qpriority]
      [child_prio=cpriority]
```

Runs on:

QNX Neutrino

Options:

-v

Display the version number for qconn, and then exit.

port=*portnum*

Select a different port to bind the daemon to. The default is 8000. If you aren't logged in as `root` when you start qconn, it can't bind to a restricted port number.

qconn_prio=*qpriority*

Set the priority at which qconn runs. The default is 10.

Use this option to increase qconn's priority if CPU-intensive programs are running at the same or a higher priority, preventing qconn's clients from receiving data. For example, getting a “Could not find target: Read timed out.” error while running the IDE's System Profiler means that the System Profiler is unable to receive data from qconn on the target.

child_prio=*cpriority*

Set the priority at which children are run. The default is 10.

Description:

The qconn daemon is a service provider that provides support, such as profiling system information, to remote IDE components. Output is based on the services invoked and is fed to the requesting IDE component on a remote host.

Copyright © 2014, QNX Software Systems Limited

If you aren't logged in as `root` when you start `qconn`, it can't start resource manager components.

The functionality comes from the service modules that are currently bound directly into the executable.

Currently, if you want to use the debugger, you must have `pdebug` installed on your system in the search path used when `qconn` is launched. You may also have `pdebug` in `/usr/bin`.

Files:

- `libtracelog.so`

Other supporting files are required, depending on the service being used. For example, the memory analysis service requires that `librcheck.so` be installed in the user's path.

Caveats:

The `qconn` daemon lets anyone run any application on your target system as the superuser. Obviously, this is a huge security risk. Don't include `qconn` on systems being deployed to customer sites.

Copyright © 2014, QNX Software Systems Limited

qcp

QNX communications protocol (QNX)

Syntax:

Send files:

```
qcp [device] se [options] src_file[,dst_file]...
    [x=index_file]...
```

Receive files:

```
qcp [device] re [options] [-f filename|-p prefix]
```

Runs on:

QNX Neutrino

Options:

device

Pathname of serial device to use. The default is the device connected to *stdin* and *stdout*.

-f *filename*

Force received files to have this name.

-F

Required to receive files on a flash filesystem that doesn't support all the POSIX file control mechanisms.

-l *logfile*

("el") Append the file transfer event to the logfile.

-m

Suppress making directories for received files.

-n

Receive only files that are newer than existing files.

-p *prefix*

Place this path prefix on the names of any received files.

-q

Copyright © 2014, QNX Software Systems Limited

Be quiet; display nothing during the transfer.

-r

Relax timing; double timeouts and quadruple retry counts.

-s *packet_size*

The size of transmitted data bursts (default is 2048 bytes).

-t

Apply today's date to received files.

-u

Unlink files that lack write permission. This allows new versions of the affected files to be received and written to disk.

-V, -v

Be verbose; display the error status while transferring files.

x=*index_file*

A list of filenames; qcp sends each file named in the list. You can specify multiple x=*index_file* options specified and intermix them with singly named files to transmit.

Description:

The qcp utility provides an error-checked file transfer protocol that *qtalk* (p. 1626) uses to transmit or receive files. This protocol is both highly efficient on packet switched networks and highly reliable through the use of 16-bit CRCs (Cyclic Redundancy Check).

The qcp utility automatically sends files with their pathnames, attributes, permissions, and date fields intact. The qcp protocol is ideally suited for use over public packet switch networks (X.25), as well as direct modem-to-modem connections. If communication errors are encountered, portions of the file are automatically resent until the far end acknowledges correct reception of the file.

If you're using qtalk to communicate with a remote system, you can both send and receive files to the remote system with the qcp utility. To cause a file to be sent from the remote system to your local system, type a command of the following form into the qtalk session you have connected to the remote system:

```
qcp se file1  file2,file3 x=file4
```

This causes the remote end to send the files to you; qtalk automatically starts qcp to receive the file. The files sent are:

Copyright © 2014, QNX Software Systems Limited

- `file1`
- `file2` (`file2` is received with the filename `file3`)
- all the files named in the index file `file4`.

You can create index files easily with the *`ls -p`* (p. 1139) command.

To send a file to the remote system, type the following command into the remote shell:

```
qcp re
```

then use the **Ctrl**–**A Ctrl**–**S** key sequence to cause `qtalk` to send a file to the remote `qcp` utility.

If you want `qcp` to send a file through an explicitly named device, use a command of the form:

```
qcp /dev/ser1 se filename
```

where `/dev/ser1` is the port to send through. To receive from a particular device, use a command similar to:

```
qcp  /dev/ser1  re
```

You can abort a `qcp` file transfer in progress by pressing either **Esc** or **Space**. In turn, `qcp` displays a prompt asking for confirmation of the action. To confirm aborting `qcp`, you enter `y`. If a remote `qcp` in a receive state must be shut down, the following control-character sequence aborts it:

```
^V^X^X
```

Note that `qcp` automatically removes any partially transferred files.

When transferring files via high speed modems, the -s 16000 option is recommended.

Copyright © 2014, QNX Software Systems Limited

qtalk

Talk over a communications line

Syntax:

```
qtalk [options] [system]
```

Runs on:

QNX Neutrino

Options:

-b [*baud*\|*data*\|*parity*\|*stop*][,...]

Change the serial port to this baud, parity, stop bits, and/or data bits. Values of 1-2 are interpreted as stop bits, 7-8 as data bits, `none`, `even`, `odd`, `mark` and `space` as parity, all other numbers as baud. Order isn't significant e.g. `-b 9600,8,n,1`

-c *hh*

Set this as the character that invokes `qtalk` commands. Default is `01` (**Ctrl**–**A**).

-D *delay*

Wait this many 1/20th sec periods before running the command specified by the -x option.

The `qtalk` utility reads any data emitted by the modem during this delay period and displays the data on your screen before the command is started.

If you use -x without this option, `qtalk` doesn't wait to start the command. As a result, if the modem emits any information once `qtalk` has emitted the dialing string, you have no way of predicting whether `qtalk` or the command sees that information.

-d *hh*

Replace ASCII rubout with this character. Default is `7f`.

-e

Enable local echoing.

-h

Copyright © 2014, QNX Software Systems Limited

Hang up the current modem line if someone dials a new system from within `qtalk` via the **Ctrl**–**A** command. By default, the current modem device is closed but the line isn't dropped.

-l ***logfile***

("el") Log a recording of the `qtalk` session in *logfile*.

-m ***modem*[,*init_string*]**

The name of the device to use. If you specify multiple -m options, `qtalk` tries them, in order, until it finds a device that isn't in use.

If you specify an *init_string*, it's emitted to the modem before anything else.

-o ***protocol*=*command***

Redefine transfer-protocol options, where *protocol* is one of the following:

This protocol:	Does a:
qcp_se	`qcp send`
qcp_re	`qcp receive`
zmodem_se	ZMODEM send
zmodem_re	ZMODEM receive
other_se	*other send*
other_re	*other receive*

and where *command* is the command that performs the file transfer. This command is run by the shell. The macro ***$MODEM*** is set to the pathname of the modem device and, in the case of a send, the macro ***$FILENAME*** is set to the filename to be transmitted.

You can disable `qtalk`'s automatic invocation of a protocol by setting *command* to a null string (""). For example, the following disables automatic ZMODEM receive:

```
qtalk -o zmodem_re=""
```

For information on default protocol commands or for more information on automatic invocation of protocols, see the section on "*Invoking qcp and ZMODEM automatically* (p. 1630)."

The other protocol lets you configure `qtalk` to use your own commands for sending and receiving files.

-P

Copyright © 2014, QNX Software Systems Limited

Ignore the parity bit of received characters.

-q

Be quiet; suppress the banner and display only a minimal prompt in command mode.

-s ***system_directory***

Instead of ***$HOME***`/.qtalk`, use this file to look up systems to dial.

-t ***xfer_protocol***

Set the current transfer protocol to *qcp* (p. 1623), `zmodem`, or `other`. (To change the command string that runs to perform the file transfer, use -o.)

-x "*command*"

Run this command after emitting the dialing string for the named system (see also -D). When the command is run, the ***MODEM*** environment variable is set to the pathname of the selected modem device.

system

The name of the system you want `qtalk` to call. The system must be defined either in ***$HOME***`/.qtalk` (or the file named by -s) or in the system-wide dialing directory, `/etc/config/qtalk`. For more information, see the description of the *d* (p. ?) (dial system) command.

Description:

The `qtalk` utility lets QNX Neutrino users communicate with other computers via a serial line that's usually connected to a modem. The destination may be another host computer, in which case `qtalk` lets you use your computer as a terminal. The `qtalk` utility also lets two QNX Neutrino users communicate and transfer files.

The `qtalk` utility sends any characters you type on the keyboard to the other system via the modem. Any characters received by the modem are displayed. In local echo mode, typed characters are echoed on the display as well as being sent over the modem.

Configuring default behavior

The `qtalk` utility lets you define new defaults for anything that you can specify with command-line options. Define these settings by creating or modifying a system (in the global and/or your personal dialing directory) called `defaults`. The first thing `qtalk` does is to look up the `defaults` dialing entry and process it, before looking at any command-line options. The `qtalk` utility looks for `defaults` first in ***$HOME***`/.qtalk` and, if not found there, then in `/etc/config/qtalk`. This allows

Copyright © 2014, QNX Software Systems Limited

new system-wide defaults to be set by a system administrator, while leaving individual users free to create their own default behavior for qtalk.

Command-line options defined in the defaults system are applied *before* the options specified in the actual command line. Thus you can override the settings in your defaults system with command-line options.

The -s command-line option doesn't change how qtalk searches for the defaults system.

Logging a session

You can log a recording of a qtalk session with the -l option:

```
qtalk -l /dev/par
```

or, if you want the log to go to the file /tmp/logfile:

```
qtalk -l /tmp/logfile
```

Using the command character

You can use a special command character to set special modes and options while you're within the qtalk environment. This special character defaults to **Ctrl–A** (^A) unless you change it with the -c option when you invoke qtalk. When you enter the command character, you're shown some current settings and are prompted to enter a command. If you enter the command character twice, a single command character is echoed to the modem. This allows you to send the command character to the remote system if you need to.

Replacing the rubout/delete character

You'll find the -d *hh* option useful when communicating with computers that have a rubout character different from the one you're used to. Many systems use the **Backspace** key (08 hex) to erase a character. QNX Neutrino systems default to the ASCII rubout character (7F hex). If you type:

```
qtalk -d 08
```

qtalk translates your **Rubout** key into backspace automatically.

Enabling flow control

Using very high-speed modems, or producing a log on slow printers or floppy disks, may cause some characters to be lost. To prevent characters from being lost in these cases, you can enable input flow control prior to invoking qtalk (see *stty* (p. 1863)). This works only if the machine sending the data supports flow control of its output.

Transferring files

The qtalk utility lets you transfer files through either of two methods:

Copyright © 2014, QNX Software Systems Limited

- You invoke the simpler method with the `w` (write) and `l` (log) commands to transfer text files to or from another system. No error checking is performed, however, so you should use this method only when communication lines are good (i.e. direct connect lines or reliable modem connections), or when the other system doesn't support any of the file transfer methods available from `qtalk`.

  To send a file to a host using write, you should first make the host capable of receiving a stream of text. You could do this with an editor or the *cat* (p. 95) utility. You can then use the write command to send the file.

  To receive a file from a remote host using the `l` (log) command, first turn on logging to the desired file (with the `l` command), then enter a command into the remote system to make it display the file (e.g. *cat* (p. 95) *filename* on a UNIX-type system.)

- The second method, involves the `s` (send) and `r` (receive) commands, providing access to more sophisticated protocols that include error checking and retransmission when transferring files to other systems.

Invoking qcp and ZMODEM automatically

The `qtalk` utility attempts to detect *qcp* (p. 1623) and `zmodem` file transfers when `qtalk` is on the receiving end. If it detects one of these, `qtalk` automatically invokes the appropriate receive command for the protocols being used. You can disable this behavior; see the *-o* (p. ?) option.

When a *qcp* (p. 1623) or `zmodem` transfer is detected, or when the `r`eceive command is given, `qtalk` runs (through a shell) one of the *protocol*_**re** commands.

The corresponding *protocol*_**se** commands (to send a file) are invoked only when the `s`end command is given; never automatically.

Here are the default protocol commands:

```
qcp_re="qcp $MODEM re"
qcp_se="qcp $MODEM se $FILENAME"
zmodem_re="rz <$MODEM >$MODEM"
zmodem_se="sz $FILENAME <$MODEM >$MODEM"
other_re=""
other_se=""
```

You can't automatically enable the *other_re* protocol. To invoke *other_re* or *other_se*, choose `s`end or `r`eceive from the command menu when the current file transfer protocol has been set to other.

Interactive commands:

To specify any of the following commands, first press the command character (usually **Ctrl**–**A**).

b (break)

Send a break over the modem. You may also send breaks by pressing the break key (**Ctrl**–**Break** on the console keyboard).

Copyright © 2014, QNX Software Systems Limited

C (command character)

Change the command character. You're prompted to enter the new command character in hex (e.g. `0x02`) or as ^*char* (e.g. `^b`).

c (change directory)

If you specify the c command, `qtalk` prompts you for a new directory name, and attempts to move to that directory. If `qtalk` moves successfully to the directory you specify, the directory becomes the current working directory (see the *pwd* (p. 1602) utility) for the duration of `qtalk`, or until you change directory again. When `qtalk` terminates, your current working directory reverts to the directory you were in when `qtalk` was invoked.

d (dial system)

If you specify the d command, `qtalk` prompts you to enter a system name. It first looks for that name in your own dialing directory (***$HOME***`/.qtalk`) and if it doesn't find it there, it looks in the system-wide dialing directory, `/etc/config/qtalk`.

If you enter a question mark (?) for the system name, the contents of both dialing directories (if they exist) are displayed and you're prompted again for a system name. To abort the dialing command and return to normal communications mode, press **Enter** without entering a name.

Dialing is implemented by looking up the system name — which can also be specified on the command line when `qtalk` is invoked — first in your ***$HOME***`/.qtalk` file, then in the `/etc/config/qtalk` file. These files share the same format:

```
system_name [dialing_string]
[   <whitespace>   command-line_options]
    …
```

In these files, you must specify the system name at the very left of the line with no whitespace before it. The dialing string is optional (this string contains commands to be sent to the modem before you're given interactive control). If you specify a dialing string, you must separate it from the system name by spaces or tabs, or both.

In dialing strings, you can specify any of the following characters; `qtalk` acts on these characters instead of sending them straight to the modem:

|

Send a carriage return.

~

Copyright © 2014, QNX Software Systems Limited

Delay for 1 second.

'

Delay for 100 milliseconds.

^

Drop DTR for 1 second (forces modem to hang up and reset).

!

End a 500-millisecond break.

o

Emit a single character represented by *o*, where *o* is an octal number of up to three digits.

\x*hh*

Same as above, but specified in hex.

Additional lines beneath the line that defines the system name and dialing string may contain additional `qtalk` command-line options to apply when talking to this system.

If you specify these additional lines, begin each one with at least one tab or space character. Note that a single command-line option can't span multiple lines. You can, however, place just one option per line.

D (delete character)

Change the delete character. You're prompted to enter the new delete character in hex (e.g. `0x08`) or as ^*char* (e.g. ^h).

e (echo)

If you specify the e command, the local echo feature is toggled. Some systems expect the "terminal" to perform local echoing (half duplex).

h (hang up)

If you specify the h command, the CTS/RTS lines are lowered for approximately 1/2 a second. This permits modems that support hardware hangup to do so.

l (log)

Begin or end logging of this session. If no log file is open, `qtalk` asks for the name of a file to log into. If logging is already in progress, it's terminated

Copyright © 2014, QNX Software Systems Limited

and the log file is closed. The `l` command records every character that is sent or received in the log file.

You can use `l` to take "snapshots" of data from a host computer at slow speed.

o (modify protocol options)

Prompts you for the protocol whose command string you wish to change (*qcp* (p. 1623), ZMODEM, `other`). The current `s`end and `r`eceive command strings are displayed and you're then prompted to enter a new command string. To set a string to null, use `""`. If you don't want to make any changes, press **Enter**.

p (parity)

Ignore parity (top bit) of received characters. If this option is already set, turn it off.

q (quit with hangup)

This executes a hangup command (see *h* (p. ?) above), then causes `qtalk` to exit.

r (invoke a receive)

Invoke the currently selected file transfer protocol to receive a file. Note that `qtalk` automatically invokes *qcp* (p. 1623) and ZMODEM if it recognizes a received startup sequence of one of these protocols.

s (send a file)

Send a file, using the currently selected file transfer protocol. This sends a file to another system running the same protocol, which is more secure than simply writing the file to the modem.

When the currently selected protocol is *qcp* (p. 1623), you can send more than one file by specifying the x=*index_file* option when `qtalk` asks for the file to send. This file contains a list of files to send, one per line. You can also specify more than one filename, separated by spaces.

The *qcp* (p. 1623) utility lets you follow the name of the file to send with the name of the destination file. Separate the two filenames with a comma. This is also true for filenames within an index file (option x=). For example:

```
Send file(s)?  file1  main.c,new_main.c
```

sends the file `file1` as `file1` and the file `main.c` with the name `new_main.c`. If you don't specify a new name, `qtalk` creates a file with the same name as the file that is sent.

Copyright © 2014, QNX Software Systems Limited

Files received by qtalk using the *qcp* (p. 1623) protocol have the same attributes and date as the file on the sending machine.

Most transfer protocols require that the modem port be configured for 8-bit data (Use the -b option to set the serial port to 8 data bits; also see the *stty* (p. 1863) command).

t (select transfer protocol)

Shows you which protocol is being used (e.g. *qcp* (p. 1623), ZMODEM) and asks you to select a new one. If you press **Enter**, no changes are made. This command controls the protocol used when you use the send and receive commands from the command menu.

x (exit)

Exit from qtalk without performing a hangup. It's good practice to use the q (quit with hangup) command for leaving qtalk, unless you really don't want to perform a hangup.

! (execute a shell command)

This command lets you execute any command from within qtalk. You'll probably find that you'll use the ! command often to:

- Execute the *ls* (p. 1139) command in order to see what files are in your current directory prior to sending some of them.
- Execute nonnative file transfer protocols (such as XMODEM or Kermit) from within qtalk if you don't want to bother setting the other protocol to the desired commands.

Examples:

Call the system named home:

```
qtalk home
```

Communicate with a machine that doesn't echo (half duplex) and expects an ASCII backspace (08 hex) to delete characters:

```
qtalk -e -d 08
```

Communicate with another system and print a hardcopy record:

```
qtalk -l /dev/par
```

Use the qcp -n option for qcp receives (i.e. receive only files that are newer than existing files):

```
qtalk -o qcp_re="qcp $MODEM re -n"
```


Copyright © 2014, QNX Software Systems Limited

Chapter 19
R

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “R”.

Copyright © 2014, QNX Software Systems Limited

racoon

IKE (ISAKMP/Oakley) key management daemon

Syntax:

```
racoon [-BdFv46] [-f configfile] [-l logfile] [-p isakmp-port]
```

Runs on:

QNX Neutrino

Options:

-4 or -6

Specifies the default address family for the sockets.

-B

Install security association(s) from the file that is specified in `/etc/racoon/racoon.conf`.

-d

Increase the debug level. Each additional `d` increases the debug level.

-F

Run `racoon` in the foreground.

-f *configfile*

Use *configfile* as the configuration file instead of the default. The default configuration file is `/etc/racoon/racoon.conf`.

-l *logfile*

Use *logfile* as the logging file instead of `syslogd`.

-p *isakmp-port*

Listen to ISAKMP key exchange on port *isakmp-port* instead of the default port number, `500`.

-v

Specifying this option causes the packet dump to be more verbose, with a higher debugging level.

Copyright © 2014, QNX Software Systems Limited

Description:

The `racoon` daemon speaks IKE (ISAKMP/Oakley) key management protocol, to establish security association with other hosts. The SPD (Security Policy Database) in the kernel usually triggers to start racoon.

Because of encryption export laws, `racoon` isn't provided in regular OS bundles. QNX Software Systems must report to the US government, identifying customers who have access to the encryption technology contained in the `racoon` daemon. If you wish to have access to this binary, you must contact your QNX sales representative, who can provide download access once approved.

Examples:

For examples showing how to configure `racoon` directives and statements, see *`/etc/racoon.conf`* (p. 1638).

Files:

`/etc/racoon/racoon.conf`

Default configuration file for `racoon`

Exit status:

`0`

Success.

`> 0`

An error occurred.

Contributing author:

OpenSSL Project

License:

This utility is based on OpenSSL Project software; for licensing information, see the Third Party License Terms List at *`http://licensing.qnx.com/third-party-terms/`*.

Copyright © 2014, QNX Software Systems Limited

/etc/racoon.conf

Configuration file for `racoon`

Name:

`/etc/racoon.conf`

Description:

This file is the configuration file for the `racoon` ISAKMP daemon. This daemon negotiates security associations for itself (ISAKMP SA, or phase 1 SA) and for kernel IPsec (IPsec SA, or phase 2 SA). The file consists of a sequence of directives and statements. Statements are enclosed by braces (`{}`). Lines beginning with # are comments.

The major parameters are listed below.

number

A hexadecimal (prefixed with `0x`) or decimal number.

string*, *path*, *file

Any string enclosed in double quotes (e.g. "*`string`*").

address

IPv6 and/or IPv4 address.

port

A TCP/UDP port number.

timeunit

One of the following:

- sec
- secs
- second
- seconds
- min
- mins
- minute
- minutes
- hour

Copyright © 2014, QNX Software Systems Limited

- hours

Path Specification

The following path specfications are allowed:

`path include` ***path***

Specifies a path to include a file, see *File Inclusion* (p. 1639), below.

`path pre_shared_key` ***file***

Specifies a file containing pre-shared key(s) for various ID(s), see *Pre-shared key File* (p. 1649), below.

`path certificate` ***path***

The `racoon` daemon will search this directory if a certificate or certificate request is received.

`path backupsa` ***file***

Specifies a file to store SA information that is negotiated by `racoon`. If you specify the -B option when you start `racoon`, the daemon will install SA(s) from this *file*.

Make sure you remove redundant data from this file to keep it to a reasonable size because `racoon` simply adds SAs (you have to do this manually).

File Inclusion

`include` ***file***

Other configuration files can be included.

Identifier Specification

This is obsolete; it must be defined at each `remote` directive.

Timer Specification

`timer {`*`statements`***`...}`**

Specifies various timer values, you can use statements such as:

`counter` ***number***

The maximum number of retries to send. The default is 5.

Copyright © 2014, QNX Software Systems Limited

interval *number timeunit*

The interval to resend, in seconds. The default time is 10 seconds.

persend *number*

The number of packets per send. The default is 1.

phase1 *number timeunit*

The maximum time it should take to complete phase 1. The default time is 15 seconds.

phase2 *number timeunit*

The maximum time it should take to complete phase 2. The default time is 10 seconds.

Listening Port Specification

listen { *statements* }

If you don't specify this directive, `racoon` will listen on all of the available interface addresses. Here are the valid statements:

isakmp *address* **[***port***]**

Tells `racoon` to listen only on *address*. The default port is 500 (specified by IANA). You can provide more than one address definition.

strict_address

Require that all addresses for ISAKMP must be bound. This statement will be ignored if you do not specify any addresses.

Remote Nodes Specifications

remote (*address* **| anonymous) [***port***] { *statements* }**

Specifies the parameters for IKE phase 1 for each remote node. The default port is 500. If `anonymous` is specified, the statements apply to all peers that do not match any other `remote` directive.

These are the valid statements:

exchange_mode (main | aggressive | base) *lc0*

Copyright © 2014, QNX Software Systems Limited

Defines the exchange mode for phase 1 when `racoon` is the initiator. Also it means the acceptable exchange mode when `racoon` is responder. More than one mode can be specified by separating them with a comma. All of the modes are acceptable. The first exchange mode is what `racoon` uses when it is the initiator.

`doi ipsec_doi`

Use `IPSEC-DOI` as specified RFC 2407. You can omit this statement.

`situation identity_only`

Use `SIT_IDENTITY_ONLY` as specified RFC 2407. You can omit this statement.

`identifier` ***idtype***

This statement is obsolete. Instead, use `my_identifier`.

`my_identifier` ***idtype*** **`...`**

Specifies the identifier sent to the remote host and the type to use in the phase 1 negotiation. `address`, `fqdn`, `user_fqdn`, `keyid` and `asn1dn` can be used as an *idtype*. They are used like this:

`my_identifier address [`*address*`]`

The type is the IP address. This is the default type if you do not specify an identifier to use.

`my_identifier user_fqdn` ***string***

The type is a `USER_FQDN` (user fully-qualified domain name).

`my_identifier fqdn` ***string***

The type is a `FQDN` (fully-qualified domain name).

`my_identifier keyid` ***file***

The type is a `KEY_ID`.

`my_identifier asn1dn [`*string*`]`

Copyright © 2014, QNX Software Systems Limited

The type is an ASN.1 distinguished name. If string is omitted, racoon will get DN from the Subject field in the certificate.

peers_identifier *idtype* **...**

This statement specifies the peer's identifier to be received. If it is not defined then racoon will not verify the peer's identifier in ID payload transmitted from the peer. If it is defined, the behavior of the verification depends on the flag of verify_identifier. The usage of idtype is the same as in my_identifier.

verify_identifier (on | off)

If you want to verify the peer's identifier, set this to on. In this case, if the value defined by peers_identifier is not the same as the peer's identifier in the ID payload, the negotiation will fail. The default is off.

certificate_type *certspec*

This specifies a certificate specification. The *certspec* variable is constructed like this:

x509 *certfile privkeyfile*

Where *certfile* is the file name of the certificate and *privkeyfile* is the file name of the secret key.

peers_certfile (dnssec | *certfile***)**

If dnssec is defined, racoon will ignore the CERT payload from the peer, and try to get the peer's certificate from DNS instead. If *certfile* is defined, racoon will ignore the CERT payload from the peer, and will use this certificate as the peer's certificate.

send_cert (on | off)

If you do not want to send a certificate for some reason, set this to off. The default is on.

send_cr (on | off)

If you do not want to send a certificate request for some reason, set this to off. The default is on.

verify_cert (on | off)

Copyright © 2014, QNX Software Systems Limited

If you do not want to verify the peer's certificate for some reason, set this to `off`. The default is `on`.

`lifetime time` ***number timeunit***

Define a lifetime which will be proposed in the phase 1 negotiations. Any proposal will be accepted, and attributes will be not proposed to the peer unless you specify them. They can be individually specified in each proposal.

`initial_contact (on | off)`

Enable this to send an `INITIAL-CONTACT` message. The default value is `on`.

`passive (on | off)`

If you do not want to initiate the negotiation, set this to `on`. The default value is `off`. This is useful for a server.

`proposal_check` ***level***

Specifies the action of lifetime length and PFS of the phase 2 selection on the responder side. The default *level* is `strict`. Here are the acceptable values for *level*:

`obey`

The responder will obey the initiator anytime.

`strict`

If the responder's length is longer than the initiator's one, the responder uses the initiator's one; otherwise, it rejects the proposal. If PFS is not required by the responder, the responder will obey the proposal. If PFS is required by both sides, and if the responder's group is not equal to the initiator's one, the responder will reject the proposal.

`claim`

If the responder's length is longer than the initiator's one, the responder will use the initiator's one. If the responder's length is shorter than the initiator's one, the responder uses its own length AND sends a `RESPONDER-LIFETIME` notify message to an initiator

Copyright © 2014, QNX Software Systems Limited

in the case of lifetime. About PFS, this directive is same as `strict`.

`exact`

If the initiator's length is not equal to the responder's one, the responder will reject the proposal. If PFS is required by both sides and if the responder's group is not equal to the initiator's one, the responder will reject the proposal.

`support_mip6 (on | off)`

If this value is set `on`, both values of ID payloads in phase 2 exchange are always used as the addresses of end-point of IPsec-SAs. The default is `off`.

`generate_policy (on | off)`

This directive is for the responder. Therefore you should set passive on in order that `racoon` only becomes a responder. If the responder does not have any policy in SPD during phase 2 negotiation, and the directive is set `on`, then `racoon` will choose the first proposal in the SA payload from the initiator, and generate policy entries from the proposal. It is useful to negotiate with the client which is allocated IP address dynamically.

Note that inappropriate policy might be installed by the initiator because the responder just installs policies as the initiator proposes. So that other communication might fail if such policies installed. This directive is ignored in the initiator case. The default value is `off`.

`nonce_size number`

Define the byte size of nonce value. The `racoon` daemon can send any value, although RFC2409 specifies that the value MUST be between 8 and 256 bytes. The default size is `16` bytes.

`proposal { `*`sub-substatements`*` }`

The *substatements* are as follows:

`encryption_algorithm `*`algorithm`*`;`

Copyright © 2014, QNX Software Systems Limited

Specify the encryption algorithm used for the phase 1 negotiation. This directive must be defined. The *algorithm* variable for oakley is one of following:

- `des`
- `3des`
- `blowfish`
- `cast128`

This statement should not be used for other transforms.

`hash_algorithm` ***algorithm***

Define the hash algorithm used for the phase 1 negotiation. This directive must be defined. The *algorithm* variable for oakley is `md5` or `sha1`.

`authentication_method` ***type***

Defines the authentication method used for the phase 1 negotiation. This directive must be defined. The *type* variable is one of:

- `pre_shared_key`
- `rsasig`
- `gssapi_krb`

`dh_group` ***group***

Define the group used for the Diffie-Hellman exponentiations. This directive must be defined. The *group* variable is one of following:

- modp768
- modp1024
- modp1536

Or you can define `1`, `2`, or `5` as the DH group number. When you want to use aggressive mode, you must define the same DH group in each proposal.

`lifetime time` ***number timeunit***

Define the lifetime of the phase 1 SA proposal. Refer to the description of lifetime directive immediately defined in the `remote` directive.

`gssapi_id` ***string***

Copyright © 2014, QNX Software Systems Limited

Define the GSS-API endpoint name, to be included as an attribute in the SA, if the `gssapi_krb` authentication method is used. If this is not defined, the default value of "ike/*hostname*" is used, where *hostname* is the FQDN of the interface being used.

Policy Specifications

The policy directive is obsolete, policies are now in the SPD. The `racoon` daemon will obey the policy configured into the kernel by `setkey`, and will construct phase 2 proposals by combining `sainfo` specifications in `/etc/racoon.conf`, and policies in the kernel.

Sainfo Specifications

`sainfo (`*`source_id destination_id`*` | anonymous) {` *`statements`* **`}`**

Defines the parameters of the IKE phase 2 (IPSec-SA establishment). The *source_id* and *destination_id* variables are constructed like this:

`address` *`address`* **`[/`** *`prefix`*`]` **`[`** *`port`* **`]`** *`ul_proto`* **or** *`idtype string`*

Means exactly the content of ID payload. This is not like a filter rule. For example, if you define `3ffe:501:4819::/48` as *source_id*, `3ffe:501:4819:1000:/64` will not match.

`pfs_group` *`group`*

Define the group of Diffie-Hellman exponentiations. If you do not require PFS, you can omit this directive. Any proposal will be accepted if you do not specify one. The *group* variable is one of following:

- `modp768`
- `modp1024`
- `modp1536`

Or you can define `1`, `2`, or `5` as the DH group number.

`lifetime time` *`number timeunit`*

Define the amount of time to be used for IPsec-SA. Any proposal will be accepted, and no unspecifed attributes will be proposed to the peer. For more informantion, see the `proposal_check` directive.

`identifier` *`idtype`*

This is obsolete, use `my_identifier` directives instead.

Copyright © 2014, QNX Software Systems Limited

my_identifier *idtype* **...**

Specifies the ID type to use for the phase 2 negotiation. The default is *address* described in the `my_identifier` directive in `remote`. This is always for the initiator, not the responder; as the responder, the `racoon` daemon can handle only the IP address type.

The `racoon` daemon does not have the list of security protocols to be negotiated. This list is passed by SPD in the kernel. Therefore, you have to define all of the potential algorithms in the phase 2 proposals, even if there is a algorithm which will not be used. These algorithms are defined by using the following three directives with a single comma as the separator.

For algorithms that can take variable-length keys, algorithm names can be followed by a key length, like `blowfish 448`. The `racoon` daemon will compute the actual phase 2 proposals by computing the permutation of the specified algorithms, and then combining them with the security protocol specified by the SPD. For example, if `des`, `3des`, `hmac_md5`, and `hmac_sha1` are specified as algorithms, we have four combinations for use with ESP, and two for AH. Then, based on the SPD settings, `racoon` will construct the actual proposals. If the SPD entry asks for ESP only, there will be 4 proposals. If it asks for both AH and ESP, there will be 8 proposals.

The kernel may not support the algorithm you have specified.

encryption_algorithm *algorithms*

Where *algorithms* may be:

- `des`
- `3des`
- `des_iv64`
- `des_iv32`
- `cast128`
- `blowfish`
- `null_enc`
- `rijndael` (used with ESP)

Copyright © 2014, QNX Software Systems Limited

authentication_algorithm *algorithms*

Where *algorithms* may be:

- des
- 3des
- des_iv64
- des_iv32
- hmac_md5
- hmac_sha1
- non_auth(used with ESP authentication and AH)

compression_algorithm *algorithms*

Where *algorithms* may be:

- deflate (used with IPComp)

Logging level

log *level*

Define logging level. The *level* variable is one of the following:

- notify (default)
- debug
- debug2

If you put too high a logging level on slower machines, IKE negotiation can fail due to timing constraint changes.

Specifying the way to pad

padding { *statements* }

The specified padding format. The following are valid statements:

randomize (on | off)

Enable using a randomized value for padding. The default is on.

randomize_length (on | off)

The pad length is random. The default is off.

maximum_length *number*

Copyright © 2014, QNX Software Systems Limited

Define a maximum padding length. If `randomize_length` is `off`, this is ignored. The default is 20 bytes.

`exclusive_tail (on | off)`

This puts the number of pad bytes minus one into the last part of the padding. The default is `on`.

`strict_check (on | off)`

This constrains the peer to set the number of pad bytes. The default is `off`.

Special directives

`complex_bundle (on | off)`

Defines the interpretation of proposal in the case of SA bundle. Normally "IP AH ESP IP payload" is proposed as "AH tunnel and ESP tunnel". The interpretation is more common to other IKE implementations, however, it allows a very limited set of combinations for proposals. With the option enabled, it will be proposed as "AH transport and ESP tunnel". The default value is `off`.

Pre-shared key File

`Pre-shared key file` defines a pair of the identifier and the shared secret key which are used at Pre-shared key authentication method in phase 1. The pair in each lines are separated by some number of blanks and/or tab characters (as in `/etc/hosts`). The key can include blanks because all of the words after the 2nd column are interpreted as a secret key.

Lines starting with # are ignored. Keys which start with `0x` are hexadecimal strings.

The file must be owned by the user ID running `racoon` (usually the privileged user), and must not be accessible by others.

Examples:

The following example shows how the `remote` directive should be configured:

```
path pre_shared_key "/usr/local/v6/etc/psk.txt" ;
remote anonymous
{
exchange_mode aggressive,main,base;
lifetime time 24 hour;
proposal {
encryption_algorithm 3des;
hash_algorithm sha1;
authentication_method pre_shared_key;
dh_group 2;
```

Copyright © 2014, QNX Software Systems Limited

```
}
}

sainfo anonymous
{
pfs_group 2;
lifetime time 12 hour ;
encryption_algorithm 3des, blowfish 448, rijndael ;
authentication_algorithm hmac_sha1, hmac_md5 ;
compression_algorithm deflate ;
}
```

The following is a sample of the file defined pre-shared key:

```
10.160.94.3 mekmitasdigoat
172.16.1.133 0x12345678
194.100.55.1 whatcertificatereally
3ffe:501:410:ffff:200:86ff:fe05:80fa mekmitasdigoat
3ffe:501:410:ffff:210:4bff:fea2:8baa mekmitasdigoat
foo@kame.net mekmitasdigoat
foo.kame.net hoge
```

Caveats:

The `racoon` daemon may not yet be able to handle some of the statements described in this document.

Copyright © 2014, QNX Software Systems Limited

racoonctl

racoon *administrative control tool*

Syntax:

```
racoonctl reload-config
racoonctl show-schedule
racoonctl [-l [-l]] show-sa [isakmp|esp|ah|ipsec]
racoonctl flush-sa [isakmp|esp|ah|ipsec]
racoonctl delete-sa saopts
racoonctl establish-sa [-u identity] saopts
racoonctl vpn-connect [-u -identity] vpn_gateway
racoonctl vpn-disconnect vpn_gateway
racoonctl show-event [-l]
racoonctl logout-user login
```

Runs on:

QNX Neutrino

Description:

You can use `racoonctl` to control the operation of the *racoon* (p. 1636) daemon.

Communication between `racoonctl` and `racoon` is done through a UNIX socket. By changing the default mode and ownership of the socket, you can allow non-`root` users to alter `racoon` behavior, so do that with caution.

The following commands are available:

`reload-config`

This should cause `racoon` to reload its configuration file.

`show-schedule`

Unknown command.

`show-sa [isakmp|esp|ah|ipsec]`

Dump the security association (SA): All the SAs if no SA class is provided, or either ISAKMP SAs, IPsec ESP SAs, IPsec AH SAs, or all IPsec SAs. Use -l to increase verbosity.

`flush-sa [isakmp|esp|ah|ipsec]`

Flush all SAs if no SA class is provided, or a class of SAs, either ISAKMP SAs, IPsec ESP SAs, IPsec AH SAs, or all IPsec SAs.

Copyright © 2014, QNX Software Systems Limited

establish-sa [-u*username*] *saopts*

Establish an SA, either an ISAKMP SA, IPsec ESP SA, or IPsec AH SA. You can use the optional -u *username* when establishing an ISAKMP SA while hybrid auth is in use. The `racoonctl` utility prompts you for the password associated with *username*, and these credentials will be used in the Xauth exchange.

The *saopts* argument has the following format:

```
isakmp {inet|inet6} src dst

{esp|ah} {inet|inet6} src/prefixlen/port dst/prefixlen/port
  {icmp|tcp|udp|any}
```

vpn-connect [-u *username*] *vpn_gateway*

This is a particular case of the `establish-sa` command. It establishes an ISAKMP SA with *vpn_gateway*.

delete-sa *saopts*

Delete an SA, either an ISAKMP SA, IPsec ESP SA, or IPsec AH SA.

vpn-disconnect *vpn_gateway*

This is a particular case of the `delete-sa` command. It kills all SAs associated with *vpn_gateway*.

show-event [-l]

Dump all events reported by `racoon`, and then quit. The -l flag causes `racoonctl` to not stop once all the events have been read, but rather to loop awaiting and reporting new events.

logout-user *login*

Delete all SAs established on behalf of the Xauth user *login*.

The following command shortcuts are available:

Shortcut	Command
rc	reload-config
ss	show-sa
sc	show-schedule
fs	flush-sa
ds	delete-sa

Copyright © 2014, QNX Software Systems Limited

Shortcut	Command
es	establish-sa
vc	vpn-connect
vd	vpn-disconnect
se	show-event
lu	logout-user

Files:

/var/racoon/racoon.sock or **/var/run/racoon.sock**

The racoon control socket.

Exit status:

0

Successful completion.

Nonzero

An error occurred.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

random

Source of secure random data

You must be `root` to start this service.

Syntax:

```
random [-hpt] [-i #]
       [-U user_name | uid[:gid[,sup_gid]*]]]
```

Runs on:

QNX Neutrino

Options:

-h

Show the usage message.

-i*#*

Use interrupt number *#* as a source for collecting random data. You may specify more than one interrupt, to a maximum of 32.

-p

Poll system information from `/proc` for random data.

-t

Use the high-performance clock as a random data source.

-U *user_name*

-U *uid*[:*gid*[,*sup_gid*]*]]

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

Copyright © 2014, QNX Software Systems Limited

Description:

The `random` service runs in the background providing a source of secure, random data suitable for encryption and security. The service builds its internal pool of random data from sources specified when it is started. These sources may include timers, interrupts, and detailed system runtime information. The service makes this random data available by providing device entries that any application can read:

- `/dev/random`
- `/dev/urandom`

These device entries provide the same functionality.

The user controls all of the sources to be used to collect random data by specifying source options on the command line.

Using interrupts as sources imposes an overhead on system performance. When using the i option, you might want to minimize the impact of this overhead by specifying only one or two interrupts from low interrupt rate devices such as disk drivers and input/serial devices.

Examples:

Start the `random` service using three PC interrupts as sources:

```
random -i12 -i14 -i15
```

From an application, read 4 bytes of random data like this:

```
int data;
int fd;

fd = open( "/dev/random", O_RDWR );
if( fd == -1 )
exit( 1 );

read( fd, &data, sizeof( data ) );

close( fd );
```

Exit status:

0

The random data is available from `/dev/random` and `/dev/urandom`.

Any other value

An error occurred; `/dev/random` and `/dev/urandom` aren't created.

Copyright © 2014, QNX Software Systems Limited

Errors:

If an error occurs, `random` sends a description of the error to *`slogger`* (p. 1807) and doesn't create `/dev/random` or `/dev/urandom`.

Contributing author:

The `random` service uses the core algorithm from the copyright-free Yarrow pseudo-random number generator (PRNG) from Counterpane Security (`http://www.counterpane.com/yarrow.html`). Bruce Schneier and John Kelsey designed the Yarrow PRNG.

Caveats:

The `random` service will not work unless you specify at least one source of random data (options -p, -t, or -i).

Copyright © 2014, QNX Software Systems Limited

ranlib

Index an archive

Syntax:

ranlib_variant [-t] [-v|-V|--version] *archive*

where *ranlib_variant* depends on the target platform, as follows:

Target platform	*ranlib_variant*
ARMv7	ntoarmv7-ranlib
x86	ntox86-ranlib

Runs on:

Linux, Microsoft Windows

Description:

The ranlib utility generates an index to the contents of an archive and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

rcp

Remote file copy

Syntax:

```
rcp [-p] source_file target_file
rcp [-p] [-r]  source_file... target_dir
```

Runs on:

QNX Neutrino

Options:

-p

Attempt to preserve (duplicate) the modification time and file mode of each source file in its corresponding target file, ignoring the *umask* (p. 2005). By default, the mode and owner of the target file are preserved if the target file already exists; otherwise the mode of the source file modified by the umask on the destination host is used.

-r

If any of the source files are directories, copy each subtree rooted at that name; in this case, the destination must be a directory.

source_file

The pathname of a file to be copied.

target_file

The pathname to which a single file is copied.

target_dir

The pathname of an *existing* directory that's to contain the output file(s).

Description:

The rcp utility copies files between machines. Each file or directory argument is either of the following:

- a remote filename of the form *rname@rhost:path*
- a local filename that has no ":" characters, or that has a "/" before any ":"

Copyright © 2014, QNX Software Systems Limited

If the specified path isn't a full pathname, it's interpreted relative to the login directory of the specified user *ruser* on *rhost*, or of your current username if no other remote username (*rname*) is specified. A path on a remote host may be quoted (using `\`, `"`, or `'`) so that the metacharacters are interpreted remotely.

The `rcp` utility doesn't prompt for passwords; it performs remote execution via *rsh* (p. 1703), and requires the same authorization.

The `rcp` utility handles third-party copies, where neither source nor target files are on the current machine.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Files:

The `rcp` utility requires the `libsocket.so` shared library.

Caveats:

The `rcp` utility doesn't always detect that the target of a copy is a file in cases where only a directory should be legal. It's also confused by any output generated by commands in a `.login` or `.profile` file on the remote host.

Copyright © 2014, QNX Software Systems Limited

readelf

Display information about an ELF binary

Syntax:

readelf_variant [*options*] *elffile*…

where *readelf_variant* depends on the target platform, as follows:

Target platform	*readelf_variant*
ARMv7	`ntoarmv7-readelf`
x86	`ntox86-readelf`

Runs on:

Linux, Microsoft Windows

Options:

See the GNU website at *http://www.gnu.org/*.

Description:

The `readelf` utility displays information about one or more ELF format object files. For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

renice

Adjust the priorities of running processes (POSIX)

Syntax:

```
renice priorityLevel [-g pgrp...]
       [-p pid...] [-u user...]
```

Runs on:

QNX Neutrino

Options:

-g *pgrp*

Renice processes in this process group.

-p *pid*

Renice this process.

-u *user*

Renice processes owned by this user.

priorityLevel

The amount to adjust the priority by. Positive numbers lower the priority by that many full priority levels, while negative numbers raise the priority.

Description:

The `renice` utility adjusts the priority of all the threads in one or more running processes. The specified *priorityLevel* is subtracted from the current priority of each selected process.

If you enter a:	`renice`:
Positive value (e.g. 2 or +2)	*Lowers* the priority of a process, making it "nicer" to other processes
Negative value (e.g. -2)	*Raises* the priority of a process, making it "meaner" to other processes

You can adjust the priority as follows:

Copyright © 2014, QNX Software Systems Limited

If you're:	You can change to any priority:
A non-`root` user	From 1 to 63
`root`	From 1 to 255

If you don't have the appropriate privileges, you can `renice` only the processes you own. You can change the range of privileged priorities with the -P option for *procnto* (p. 1586).

Examples:

Lower the priority of process 768 by 2:

```
renice 2 -p 768
```

Exit status:

0

Successful completion.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

/etc/resolv.conf

Resolver configuration file

Name:

/etc/resolv.conf

Description:

The resolver library routines provide access to the Internet Domain Name System (DNS). When these routines are first invoked by a process, they read information contained in the resolver configuration file. This file contains a list of keywords with user-specified values that provide various types of resolver information.

This file is optional. If it isn't present:

- the resolver library routines look in /etc/hosts only to resolve the hostname
- the domain name is determined from the hostname
- the domain search path is constructed from the domain name.

Overriding /etc/resolv.conf

You can use the following *confstr()* configuration strings to override the data contained in /etc/resolv.conf:

_CS_DOMAIN

The domain name, without any keyword. For example:

```
my.domain
```

_CS_RESOLVE

The contents of the resolv.conf file, except that the configuration string:

- doesn't include the domain name
- can't include spaces; separate the keywords by underscores

For example:

```
nameserver_209.226.137.53
```

The socket library uses the following search order to locate the resolver data:

1. *confstr()* configuration strings
2. resolv.conf.*hostname*
3. resolv.conf

Copyright © 2014, QNX Software Systems Limited

Utilities such as *dhcp.client* (p. 544) and *pppd* (p. 1560) can optionally set the configuration strings.

Keywords

The keyword and its associated value must appear on a single line. The line must start with the keyword (e.g. `nameserver`) followed by whitespace and the value.

The `domain` and `search` keywords are mutually exclusive. If more than one instance of these keywords is present, the last instance overrides any others.

`nameserver`

The Internet address (in dot notation) of a name server that the resolver should query. Up to `MAXNS` (currently 3) name servers may be listed, one per keyword. If multiple server entries are present, the resolver library queries them in the order listed. If no server entries are present, the default is to use the name server on the local machine. (The algorithm used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made).

`domain`

The local domain. Most queries for names within this domain can use short names relative to the local domain. If no domain entry is present, the domain is determined from the local hostname returned by *gethostname()*; the domain part is taken to be everything after the first dot. If the hostname doesn't contain a domain part, the root domain is assumed.

`search`

The search list used for looking up hostnames. The search list is normally determined from the local domain name. By default, it begins with the local domain name, then with successive parent domains that have at least two components in their names.

You can override the default list by specifying the desired domain search path and by following the `search` keyword with the names. Most resolver queries are attempted using each component of the search path in turn until a match is found.

This process may be slow and generates a lot of network traffic if the servers for the listed domains aren't local. Queries time out if no server is available for one of the domains.

The search list is currently limited to six domains with a total of 256 characters.

nocache

By default, the `resolv.conf` data is parsed at application startup only. It is not checked again. Specifying “nocache on” will cause the `resolv.conf` data to be parsed at every lookup. If you wish to invalidate the cache at a specific time, it would be better to call *res_init()* directly, or turn off the _res.options flag `RES_INIT`.

Copyright © 2014, QNX Software Systems Limited

For more information, see:

- *TCP/IP Network Administration*
- *DNS and BIND* by Paul Albitz and Cricket Liu, O'Reilly & Associates (ISBN 1-56592-010-4)

Copyright © 2014, QNX Software Systems Limited

~/.rhosts

Individual users' list of trusted remote users

Name:

`~/.rhosts`

Description:

The `~/.rhosts` and *`/etc/hosts.equiv`* (p. 947) files provide the "remote authentication" database for the *`rcp`* (p. 1658), *`rlogin`* (p. 1669), and *`rsh`* (p. 1703) commands and the *rcmd()* function. These files bypass the standard password-based user authentication mechanism. They specify remote hosts and users that are considered *trusted* (i.e. are allowed to access the local system without supplying a password):

- on a system-wide basis (`/etc/hosts.equiv`)
- by an individual user (`~/.rhosts`).

The file permissions for the `~/.rhosts` file must be as follows or its contents will be ignored:

- it must be owned by `root` or the user
- it cannot be writable by anyone other than the owner (e.g. `rw-r--r--`)

The *ruserok()* function sets the effective userid to that of the remote user, but doesn't change the effective group ID. The user must have search permissions for the directories contained in the pathname of an `.rhosts` file (i.e. if the file resides in `/home/user/.rhosts`, the user must have search permissions for `/home/user/`).

The library routine *ruserok()* (see also *rcmd()*) performs the remote authentication. It determines whether a particular remote user from a particular remote host is allowed to access the local system as a (possibly different) particular local user:

- For non-`root` users, this routine checks `/etc/hosts.equiv`, and then the `.rhosts` file in the home directory of the local user attempting access.
- For `root`, access is handled as a special case to help maintain system security; only `root`'s `.rhosts` file is checked.

The *`rlogind`* (p. 1671) daemon doesn't allow `root` to log in without a password. When `rsh` is specified without command options, `rlogind` (not `rshd`) is invoked on the remote side.

Copyright © 2014, QNX Software Systems Limited

If the remote authentication fails, `rcp` and `rsh` fail, but `rlogin` falls back to the standard password-based login procedure.

Both files are formatted as a list of one-line entries of the form:

```
hostname [username]
```

where *hostname* must be the fully qualified domain name (FQDN) of the host, not one of its aliases.

The entries in these files are either *positive*, to explicitly allow access without a password, or *negative*, to deny it. Authentication succeeds as soon as a matching positive entry is found, but fails when a matching negative entry is found, or if no matching entries are found in either file. Therefore, the order of entries is important: if the files contain both matching positive and negative entries, the entry that appears first prevails.

Positive entries

Positive entries take these forms:

hostname

All users from the named host are trusted and may access the system with the same user name as they have on the remote system. You can use this form in both `/etc/hosts.equiv` and individual users' `.rhosts` files.

hostname username

The meaning of this form depends on which file it's in:

- `.rhosts` file in a local user's home directory — the named user from the named host can access the system as that local user.
- `/etc/hosts.equiv` — the named remote user can access the system as *any* local user.

You can use the special character "+" as a wild card in place of either *hostname* or *username* to match any host or user:

+

Any user from any remote host can access the system, with the same username.

+ username

The named user from any remote host can access the system.

hostname +

Any user from the named host can access the system as the local user.

Copyright © 2014, QNX Software Systems Limited

Negative entries

Negative entries have a "`-`" character preceding either the *hostname* or *username* field. For example:

hostname -username

Deny access to the named user if they attempt to access your system from the named host without providing a password.

Caveats:

Use extreme caution in `/etc/hosts.equiv` with positive entries that include a username field (either an individual named user, a netgroup, or "+" sign). Because `/etc/hosts.equiv` applies system-wide, these entries allow one or a group of remote users to access the system as any local user without providing a password. This can be a security hole.

The file permissions for the `~/.rhosts` file must be as follows or its contents will be ignored:

- it must be owned by `root` or the user
- it cannot be writable by anyone other than the owner (e.g. `rw-r--r--`)

Copyright © 2014, QNX Software Systems Limited

rlogin

Remote login

Syntax:

```
rlogin [-8dE] [-e char] [-l username] host
```

Runs on:

QNX Neutrino

Options:

-8

Allow an eight-bit input data path at all times. Without this option, parity bits are stripped whenever the remote side's stop and start characters are ^S and ^Q.

-d

Turn on socket debugging on the TCP sockets used for communication with the remote host. See *setsockopt()*.

-E

Stop any character from being recognized as an escape character. When used with -8, this provides a completely transparent connection.

-e *char*

Use the specified character as the escape character (default is ~). You can specify this as a literal character or as an octal value in the form *nnn*.

-l *username*

Log in with this user ID instead of the current one.

host

The official name, an alias, or the Internet address of a remote host.

Description:

The `rlogin` utility starts a terminal session on the specified remote host. To validate the login ID, `rlogin` uses the standard Berkeley `rhosts` authorization mechanism.

Copyright © 2014, QNX Software Systems Limited

Once you're connected, typing:

```
escape_char .
```

disconnects you from the remote host. By default, the tilde (~) character is the escape character.

All echoing takes place at the remote site, so that (except for delays) the `rlogin` is transparent. Flow control via ^S and ^Q and flushing of input and output on interrupts are handled properly.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Files:

The `rlogin` utility requires the `libsocket.so` shared library.

Environment variables:

TERM

Determines the user's terminal type.

Copyright © 2014, QNX Software Systems Limited

rlogind

Remote login daemon

You must be `root` to start this daemon.

Syntax:

```
rlogind [-aln]
```

Runs on:

QNX Neutrino

Options:

-a

Ask hostname for verification.

-l

Prevent any authentication based on the user's `.rhosts` file, unless the user is logging in as the superuser.

-n

Disable keepalive messages.

Description:

The `rlogind` daemon is the server for the `rlogin` utility. The daemon provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

The `rlogind` daemon is started when *`inetd`* (p. 977) receives a service request at the port indicated by the *login* entry (`inetd` listens for service requests specified in the *`inetd.conf`* (p. 979) file at a port defined in the *`services`* (p. 1744) file). Note that the descriptions in the default `inetd.conf` file are commented out; uncomment the ones that you want to use. The following protocol is initiated:

1. The server checks the client's source port. If the port isn't in the range `512-1023`, the server aborts the connection.
2. The server checks the client's source address and requests the corresponding hostname (see *gethostbyaddr()*, the */etc/hosts* (p. 946) file, and *named* (p. 1332)).

Copyright © 2014, QNX Software Systems Limited

If the hostname can't be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the -a option is given, `rlogind` requests the addresses for the hostname and verifies that the name and address correspond. Normal authentication is bypassed if the address verification fails.

Once the source port and address have been checked, `rlogind` proceeds with the authentication process described in `rshd`. It then allocates a pseudo terminal, and manipulates file descriptors so that the slave half of the pseudo terminal becomes the standard input, standard output, and standard error for a login process.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of `rlogin`. In most cases, "`^S/^Q`" facilities are provided to propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the ***TERM*** environment variable. The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages, which are enabled unless -n is given, allow sessions to be timed out if the client crashes or becomes unreachable.

Diagnostics

All initial diagnostic messages are indicated by a leading byte with a value of `1`, after which any network connections are closed. If there are no errors before `login` is invoked, a `NULL` byte is returned as an indication of success.

`Try again`

A *fork* by the server failed.

Caveats:

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but useful in an "open" environment.

Copyright © 2014, QNX Software Systems Limited

rm

Remove files (POSIX)

Syntax:

```
rm [-Rfir] [-d] [-l n] [-v] file...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-d

(QNX Neutrino extension) If the -R option is specified, remove the files but leave the directory tree intact (i.e. no *rmdir* (p. 1675) is performed).

-f

Force each specified file to be removed without prompting for confirmation.

-i

Be interactive; request confirmation before removing each existing file.

-l *n*

("el") (QNX Neutrino extension) If the -R option is specified, recurse only *n* levels down a directory tree.

-r

Equivalent to the -R option (below).

-R

Recursively remove files and subdirectories under directories given as arguments. This process removes the directory and the entire file tree under it.

Use the -R option with care, as it removes directories, subdirectories, and files. Using the -i option with -R adds a measure of safety by turning on interactive prompting before each file or directory is removed.

-v

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino extension) Be verbose; print files and directories as they're removed.

file

The pathname of a file to be removed.

Description:

The `rm` utility removes each specified file from a directory.

By default, `rm` refuses to remove any file that names a directory. This may be overridden with the -R or -r options. In any case, `rm` always refuses to remove the current working directory.

If a *file* operand has been specified but doesn't exist and the -f option hasn't been given, a message is written to the standard error output. If -f has been given, the error message isn't written. In either case, `rm` goes on to any remaining files specified on the command line.

The `rm` utility doesn't necessarily remove the file itself. A file may have more than one link; that is, it may be known by more than one name in the filesystem (see the *ln* (p. 1114) utility for information on creating links). The `rm` utility breaks one such link; it dissociates the file from one name. If this is the only link, the file data becomes inaccessible and the file space is returned to the system for reuse. Otherwise, the data remains accessible via other names.

Examples:

Remove the `a.out` and `core` files:

```
rm a.out core
```

Remove the directory `junk` and all its contents, without prompting:

```
rm -Rf junk
```

Exit status:

0

All the named files were removed.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

rmdir

Remove directories (POSIX)

Syntax:

```
rmdir [-p] dir...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-p

Remove the entire directory path (consisting only of directories that are empty except for the named pathname components) of the directories specified on the command line. (See example.)

dir

The pathname of an empty directory to be removed.

Description:

The `rmdir` utility removes the directory specified by each *dir* operand, provided the directory is empty.

The `rmdir` utility processes directories in the order they're specified on the command line. If you specify a parent directory, you should specify its subdirectory *before* the parent directory. That way, the parent directory will be empty when the `rmdir` utility tries to remove it.

Examples:

Remove the subdirectory `oldfiles` from the `/home/fred` directory:

```
rmdir /home/fred/oldfiles
```

Remove the directory path `dir1/dir2/dir3`:

```
rmdir -p dir1/dir2/dir3
```

For this command, `rmdir` first removes `dir1/dir2/dir3`. If this is successful, it then removes `dir1/dir2`. If this is also successful, it removes `dir1`.

Exit status:

Copyright © 2014, QNX Software Systems Limited

0

Each directory specified by a *dir* operand referred to an empty directory and was removed successfully.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

rndc

Name server control utility

Syntax:

```
rndc [-b source-address] [-c config-file] [-k key-file] [-s
server]
     [-p port] [-V] [-y key_id] {command}
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?rndc++NetBSD-5.0* in the NetBSD documentation.

Description:

The `rndc` utility controls the operation of a name server. Itcommunicates with the name server over a TCP connection, sending commands authenticated with digital signatures. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?rndc++NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

rndc-confgen

Key-generation tool for `rndc`

Syntax:

```
rndc-confgen [-a] [-b keysize] [-c keyfile] [-h] [-k keyname]
             [-p port] [-r randomfile] [-s address]
             [-t chrootdir] [-u user]
```

Runs on:

QNX Neutrino

Options:

See *http://netbsd.gw.com/cgi-bin/man-cgi?rndc-confgen+8+NetBSD-5.0* in the NetBSD documentation.

Description:

The `rndc-confgen` utility generates configuration files for `rndc`. You can use it as a convenient alternative to writing the `rndc.conf` file and its `controls` and `key` statements by hand. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?rndc-confgen+8+NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

rndc.conf

Configuration file for rndc

Name:

rndc.conf

Description:

The rndc.conf file is the configuration file for rndc. For more information, see *http://netbsd.gw.com/cgi-bin/man-cgi?rndc.conf+5+NetBSD-5.0* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

route

Manually manipulate the routing tables

Syntax:

```
route [-f] [-n] [-q] [-v] command { [[modifiers] args] }
```

Runs on:

QNX Neutrino

Options:

-f

Remove all routes (as per flush). If used in conjunction with the add, change, delete, or get commands, `route` removes the routes before performing the command.

-n

Don't print host and network names symbolically when reporting actions. (The process of translating between symbolic names and numerical equivalents can be quite time consuming, and may require correct operation of the network; thus it may be expedient to forgo this, especially when attempting to repair networking operations.)

-q

Be quiet: suppress all output.

-v

Be verbose: print additional details.

***command* [[*modifiers*] *args*]**

Valid commands are: add, change, delete, flush, and show. See the "Description" section for the syntax and description of each command.

Description:

You use the `route` utility to manually manipulate the network routing tables. Because the routing tables are usually taken care of by the `routed` daemon, you rarely need to use this utility.

***command* options**

Copyright © 2014, QNX Software Systems Limited

The `route` utility accepts the following commands: add, change, delete, flush, get, monitor, and show.

Here's the syntax and the description for each command:

[-n] add [-netl-host] ***destination gateway***

Add a route.

[-n] change [-netl-host] ***destination gateway***

Change aspects of a route (such as its gateway).

[-n] delete [-netl-host] ***destination gateway***

Delete a specific route.

[-n] flush [*family***]**

(INET and INET6 only) Flush the routing tables of all gateway entries. If you want to delete only routes having destinations with addresses in a specified family, specify `INET` or `INET6` as the *family* variable.

[-n] get [-netl-host] ***destination gateway***

Look up and display the route for a destination.

[-n] monitor

Report changes to the routing information on a continuing basis.

[-n] show

Display route table (similar to `netstat -r`).

destination

The destination host or network.

gateway

The next-hop gateway that packets should be addressed to.

If the keyword, default, or the network address, 0.0.0.0, is specified, then *all* packets sent to a remote network that's not defined in the routing tables, are sent to the specified gateway.

If you have an Internet Service Provider (ISP), packets sent to hosts on the Internet are sent to a gateway provided by the ISP. See the defaultroute option in *pppd* (p. 1560).

Copyright © 2014, QNX Software Systems Limited

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. Specifying the optional keywords -net and -host force the destination to be interpreted as a network or a host, respectively.

If the *destination* has a "local address part" of `INADDR_ANY`, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, the route is assumed to be to a host. For example:

This *destination*:	Is interpreted as:
`128.32`	`-host 128.0.0.32`
`128.32.130`	`-host 128.32.0.130`
`-net 128.32`	`128.32.0.0`
`-net 128.32.130`	`128.32.130.0.`

If the route is via an interface rather than via a gateway, you should specify the -interface modifier; the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

You can use the optional -netmask modifier to specify an additional address parameter that's interpreted as a network mask. You can use this like an OSI ESIS redirect with the netmask option, or to manually add subnet routes with netmasks different from that of the implied network interface (as would otherwise be communicated using the OSPF or ISIS routing protocols). After -netmask, enter the address parameter you want interpreted as the network mask.

You can override the implicit network mask generated in the INET case by placing this option after the *destination* parameter.

Similarly, you can use the -prefixlen modifier for IPv6.

Routes have associated flags which influence operation of the protocols when sending to destinations matched by the routes. These flags may be set (or sometimes cleared) by indicating the following corresponding modifiers:

-cloning

RTF_CLONING — generates a new route on use

-xresolve

RTF_XRESOLVE — emit mesg on use (for external lookup)

-iface

~RTF_GATEWAY — destination is directly reachable

-static

Copyright © 2014, QNX Software Systems Limited

RTF_STATIC — manually added route

-nostatic

~RTF_STATIC — pretend route added by kernel or daemon

-reject

RTF_REJECT — emit an ICMP unreachable when matched

-blackhole

RTF_BLACKHOLE — silently discard pkts (during updates)

-proto1

RTF_PROTO1 — set protocol specific routing flag #1

-proto2

RTF_PROTO2 — set protocol specific routing flag #2

-llinfo

RTF_LLINFO — validly translates proto addr to link addr

The optional modifiers:

- -expire
- -hopcount
- -mtu
- -recvpipe
- -rtt
- -rttvar
- -sendpipe
- -ssthresh

provide initial values to metrics maintained in the routing entry. To lock any of these modifiers, precede the modifier with the -lock meta-modifier; you can also specify the -lockrest meta-modifier to lock all ensuing metrics.

All symbolic names specified for a destination or gateway are looked up first as a hostname using *gethostname()*. If this lookup fails, *getnetbyname()* is then used to interpret the name as that of a network.

The `route` utility uses a routing socket and the new message types `RTM_ADD`, `RTM_DELETE`, and `RTM_CHANGE`. As such, only the superuser may modify the routing tables.

Diagnostics

`add [host | network ] %s: gateway %s flags %x`

Copyright © 2014, QNX Software Systems Limited

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl()* call. If the gateway address used isn't the primary address of the gateway—the first one returned by *gethostname()* — the gateway address is printed numerically as well as symbolically.

`delete [ host &| network ] %s: gateway %s flags %x`

As above, but when deleting an entry.

`%s %s done`

A routing table entry is being deleted by the flush command.

`Network is unreachable`

An attempt to add a route failed because the gateway listed wasn't on a directly connected network. The next-hop gateway must be given.

`not in table`

A delete operation was attempted for an entry not present in the tables.

`routing table overflow`

An add operation was attempted, but the system was low on resources and couldn't allocate memory to create the new entry.

`Permission denied`

The attempted operation is privileged. Only `root` may modify the routing tables. These privileges are enforced by the kernel.

License:

This utility is based on copyright software of the Regents of the University of California and of Christos Zoulas; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Copyright © 2014, QNX Software Systems Limited

route6d

RIP6 routing daemon

Syntax:

```
route6d [-aDdhlqSs] [-A prefix/preflen,if1[,if2...]]
        [-L prefix/preflen,if1[,if2...]]
        [-N if1[,if2...]] [-O prefix/preflen,if1[,if2...]]
        [-R routelog] [-T if1[,if2...]] [-t tag]
```

Runs on:

QNX Neutrino

Options:

-A *prefix*/*preflen*,*if1*[,*if2*...]

Allow routes to be aggregated, where *prefix* specifies the prefix, and *preflen* the prefix length, of the aggregated route. When advertising routes, `route6d` filters specific routes covered by the aggregate, and advertises the aggregated route (*prefix*/*preflen*) to the interfaces specified in the comma-separated interface list, *if1*[,*if2*...]. `route6d` creates A static route to *prefix*/*preflen* with the RTF_REJECT flag is created in the kernel routing table.

-a

Enable the aging of the statically-defined routes. Remove any statically-defined routes unless the corresponding updates arrive as if the routes are received at the startup of `route6d`.

-D

Enable extensive output of debugging messages and instruct `route6d` to run in the foreground (don't become a daemon).

-d

Enable the output of debugging message and instruct `route6d` to run in the foreground (don't become daemon).

-h

Disable the *split-horizon* processing.

-L *prefix*/*preflen*,*if1*[,*if2*...]

Copyright © 2014, QNX Software Systems Limited

Filter incoming routes from interfaces *if1*,[*if2*...] and accept incoming routes that are in *prefix*/*preflen*. If you'd like to accept any route, don't specify this option.

If multiple -L options are specified, any routes that match one of the options is accepted. If `::/0` is specified, it's treated as the default route, not "any route that has longer *prefix* length than, or equal to 0." For example, `route6d` accepts the default route and routes in the 6bone test address, but in no others, if you specify:

```
-L 3ffe::/16,if1 -L ::/0,if1
```

-l

Exchange site-local routes as well. By default, they aren't exchanged because of safety reasons. The semantics of the site-local address space is rather vague (the specification is still being worked on), and there's no good way to define site-local boundaries. It must not be used on site-boundary routers, since this option assumes that all interfaces are in the same site.

-N *if1*[,*if2*...]

Don't listen to, or advertise, routes from or to interfaces specified by *if1*,[*if2*...].

-O *prefix*/*preflen*,*if1*[,*if2*...]

Restrict route advertisements toward interfaces specified by *if1*,[*if2*...] and only advertise routes that match *prefix*/*preflen*.

-q

Listen-only mode. Don't send advertisements.

-R *routelog*

Log the route change (add/delete) to the file *routelog*.

-S

Same as -s, except that the *split-horizon* rule doesn't apply.

-s

Advertise the statically defined routes that exist in the kernel routing table when `route6d` invoked. Announcements obey the regular *split-horizon* rule.

-T *if1*[,*if2*...]

Advertise only default routes toward *if1*,[*if2*...].

Copyright © 2014, QNX Software Systems Limited

-t *tag*

Attach this route tag to originated route entries. You can specify *tag* in decimal, octal (prefixed by `0`), or hexadecimal (prefixed by `0x`).

Description:

The `route6d` utility is a routing daemon which supports Routing Information Procotol (RIP) over IPv6.

When a `SIGINT` or `SIGUSR1` signal is received, `route6d` dumps the current internal state into `/var/run/route6d_dump`.

The `route6d` utility uses IPv6 advanced API, defined in *RFC2292*, for communicating with peers using link-local addresses.

Internally `route6d` embeds the interface identifier into bits 32 to 63 of link-local addresses (`fe80`::*xx* and `ff02`::*xx*) so they'll be visible on the internal state dump file (`/var/run/route6d_dump`).

The routing table manipulation differs from IPv6 implementation to implementation. Currently `route6d` obeys WIDE Hydrangea/KAME IPv6 kernel. Currently, `route6d` doesn't reduce the rate of the triggered updates when consecutive updates arrive.

For more information, see G. Malkin, and R. Minnear, *RIPng for IPv6*, *RFC2080*, January 1997.

Files:

`/var/run/route6d_dump`

Dump internal state on `SIGINT` or `SIGUSR1`.

Copyright © 2014, QNX Software Systems Limited

routed

Network RIP and router discovery routing daemon

Syntax:

```
routed [-Adghmqstv] [-F net [/mask[,metric]]]
       [-P parms] [-T tracefile]
```

Runs on:

QNX Neutrino

Options:

-A

Don't ignore RIPv2 authentication if we don't care about RIPv2 authentication. This option is required for conformance with *RFC 1723*. However, it makes no sense and breaks using RIP as a discovery protocol to ignore all RIPv2 packets that carry authentication when this machine doesn't care about authentication.

-d

Don't run in the background (for interactive use only).

-F *net*[*/mask*][*,metric*]

Minimize routes in transmissions via interfaces with addresses that match *net*/*mask*, and synthesizes a default route to this machine with the *metric*. The intent is to reduce RIP traffic on slow, point-to-point links such as PPP links by replacing many large UDP packets of RIP information with a single, small packet containing a "fake" default route. If *metric* is absent, a value of 14 is assumed to limit the spread of the "fake" default route. This is a dangerous feature that when used carelessly can cause routing loops. Notice also that more than one interface can match the specified network number and mask. See also -g.

-g

This option, which is used on internetwork routers to offer a route to the "default" destination, is typically used on a gateway:

- to the Internet
- that uses another routing protocol whose routes aren't reported to other local routers.

Copyright © 2014, QNX Software Systems Limited

It's equivalent to -F 0/0,1 and exists for historical reasons. Because a metric of 1 is used, it's very likely that you'll create chaos with a routing loop rather than solve your problem. We recommend that you use -P pm_rdisc on the command line, or add pm_rdisc to your */etc/gateways* (p. 863) file. Because a larger metric is used, the likelihood of spreading a potentially dangerous default route is reduced.

-h

Don't advertise host or point-to-point routes, provided there's a network route going in the same direction. This option is a limited kind of aggregation, and is useful on gateways to Ethernets that have other gateway machines connected with point-to-point links.

-m

Advertise a host or point-to-point route to a machine's primary interface. It's useful on multi-homed machines such as NFS servers, and should be used only when the cost of the host routes it generates is justified by the popularity of the server. Because there's more than one interface, it's effective only when the machine is supplying routing information. If specified with -q, the host route is advertised.

-P ***parms***

Equivalent to adding the *parms* parameter line to */etc/gateways* (p. 863).

-q

Don't supply routing information (opposite of the -s option). The default if only one interface is present.

-s

Force `routed` to supply routing information. The default if more than one network interfaces are present and the TCP/IP stack is set to forward between interfaces (*ipforwarding*=1; see *sysctl* (p. 1875)).

-T ***tracefile***

Increase the debugging level to at least 1 and append debugging information to the trace file. Due to security concerns, it's not recommended that you routinely run `routed` when tracing is directed to a file.

-t

Increase the debugging level so that more information is logged to the tracefile specified with -T, or to standard output. The debugging level can

Copyright © 2014, QNX Software Systems Limited

be increased or decreased with the `SIGUSR1` or `SIGUSR2` signals or with `rtquery`.

-v

Display and log the version of the daemon.

logfile

The name of file in which `routed`'s actions are to be logged. This log contains information about any changes to the routing tables; if -t isn't specified to trace all packets, the log also maintains a history of recent messages sent and received that are related to the changed route.

Any other argument supplied is interpreted as the name of a file in which the actions of `routed` should be logged. It's better to use -T instead of appending the name of the trace file to the command.

Description:

The `routed` daemon is invoked at boot time to manage the network routing tables. It uses Routing Information Protocol (RIP), RIPv1 (*RFC 1058*), RIPv2 (*RFC 1723*), and Internet Router Discovery Protocol (*RFC 1256*) to maintain the kernel routing table. The RIPv1 protocol is based on the reference 4.3BSD daemon.

It listens on the UDP socket for the `route` service for Routing Information Protocol packets. It also sends and receives multicast Router Discovery ICMP messages. If the host is a router, `routed` periodically supplies copies of its routing tables to any directly connected hosts and networks. It also advertise or solicits default routes using Router Discovery ICMP messages.

When started (or when a network interface is later turned on), `routed` uses an AF_ROUTE address family facility to find those directly connected interfaces configured into the system and marked "up." It adds necessary routes for the interfaces to the kernel routing table. Soon after being first started, and provided there is at least one interface on which RIP hasn't been disabled, `routed` deletes all pre-existing non-static routes in kernel table. Static routes in the kernel table are preserved and included in RIP responses if they have a valid RIP metric.

If more than one interface is present (not counting the loopback interface), it is assumed that the host should forward packets among the connected networks. After transmitting a RIP `request` and Router Discovery Advertisements or Solicitations on a new interface, the daemon enters a loop, listening for RIP request and response and Router Discovery packets from other hosts.

Copyright © 2014, QNX Software Systems Limited

When a `request` packet is received, `routed` formulates a reply based on the information maintained in its internal tables. The `response` packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16 or greater is considered "infinite"). Advertised metrics reflect the metric associated with interface (see `ifconfig`), so setting the metric on an interface is an effective way to steer traffic.

Responses don't include routes with a first hop on the requesting network to implement in part *split-horizon*. Requests from query programs such as `rtquery` are answered with the complete table.

The routing table maintained by the daemon includes space for several gateways for each destination to speed recovery from a failing router. RIP `response` packets received are used to update the routing tables provided they are from one of the several currently recognized gateways or advertise a better metric than at least one of the existing gateways.

When an update is applied, `routed` records the change in its own tables and updates the kernel routing table if the best route to the destination changes. The change in the kernel routing table is reflected in the next batch of `response` packets sent. If the next response isn't scheduled for a while, a *flash update* response containing only recently changed routes is sent.

In addition to processing incoming packets, `routed` also periodically checks the routing table entries. If an entry hasn't been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed until the route has been advertised with an infinite metric to insure the invalidation is propagated throughout the local internet. This is a form of *poison reverse*. Routes in the kernel table, that are added or changed as a result of ICMP redirect messages, are deleted after a while to minimize black-holes. When a TCP connection suffers a timeout, the kernel tells `routed`, which deletes all redirected routes through the gateway involved, advances the age of all RIP routes through the gateway to allow an alternate to be chosen, and advances of the age of any relevant Router Discovery Protocol default routes.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. These RIP responses are sent to the broadcast address on nets that support broadcasting, to the destination address on point-to-point links, and to the router's own address on other networks. If RIPv2 is enabled, multicast packets are sent on interfaces that support multicasting.

If no response is received on a remote interface, if there are errors while sending responses, or if there are more errors than input or output (see `netstat`), then the cable or some other part of the interface is assumed to be disconnected or broken, and routes are adjusted appropriately.

Copyright © 2014, QNX Software Systems Limited

The *Internet Router Discovery Protocol* is handled similarly. When the daemon is supplying RIP routes, it also listens for Router Discovery Solicitations and sends Advertisements. When it is quiet and listening to other RIP routers, it sends Solicitations and listens for Advertisements. If it receives a good Advertisement and it's not multihomed, it stops listening for broadcast or multicast RIP responses. It tracks several advertising routers to speed recovery when the currently chosen router dies. If all discovered routers disappear, the daemon resumes listening to RIP responses. It continues listening to RIP while using Router Discovery if multihomed to ensure all interfaces are used.

The Router Discovery standard requires that advertisements have a default "lifetime" of 30 minutes. That means should something happen, a client can be without a good route for 30 minutes. It is a good idea to reduce the default to 45 seconds using

```
-P rdisc_interval=45
```

on the command line, or

```
rdisc_interval=45
```

in the */etc/gateways* (p. 863) file.

While using Router Discovery (which happens by default when the system has a single network interface and a Router Discover Advertisement is received), there is a single default route and a variable number of redirected host routes in the kernel table. On a host with more than one network interface, this default route will be via only one of the interfaces. Thus, multi-homed hosts running with -q might need no_rdisc described below.

See the pm_rdisc facility described below to support "legacy" systems that can handle neither RIPv2 nor Router Discovery.

By default, neither Router Discovery advertisements nor solicitations are sent over point to point links (e.g. PPP). The netmask associated with point-to-point links (such as PPP with the `IFF_POINTOPOINT` flag) is used by `routed` to infer the netmask used by the remote system when RIPv1 is used.

The `routed` daemon supports the notion of "distant" *passive* or *active* gateways. When `routed` is started, it reads */etc/gateways* (p. 863) to:

- find distant gateways which may not be located using only the information from a routing socket
- discover if some of the local gateways are passive
- obtain other parameters.

For more information, see:

- the */etc/gateways* (p. 863) file
- `gated` (an unsupported version is available on Foundry27)

Copyright © 2014, QNX Software Systems Limited

- *Internet Transport Protocols*, XSIS 028112, Xerox System Integration Standard.

Files:

/etc/gateways **(p. 863)**

List of the distant gateways which may not be located using only the information from a routing socket.

Copyright © 2014, QNX Software Systems Limited

/etc/rpc

RPC program number database

Name:

/etc/rpc

Description:

The rpc file contains user-readable names that can be used in place of rpc program numbers. Each line has the following information:

```
name_of_server_for_rpc_program   rpc_program_number  aliases
```

Items are separated by any number of **blank**s or **tab**s, or both. A pound sign (#) in a line indicates the beginning of a comment — any characters after a #, up to the end of the line, aren't interpreted by routines that search the file.

Copyright © 2014, QNX Software Systems Limited

rpcbind

Map RPC program numbers into universal addresses

Syntax:

```
rpcbind [-dilLs]
```

Runs on:

QNX Neutrino

Options:

-d

Run in debug mode. In this mode, it doesn't fork when it starts. Also, it prints additional information during operation, or aborts on certain errors. Name-to-address translation consistency checks are also shown in detail.

-i

Make `rpcbind` insecure. You need this option to switch to nonsecure mode. This option allows SET and UNSET from any host. Normally `rpcbind` accepts these requests only from the loopback interface for security reasons. This change is necessary for programs that were compiled with earlier versions of the RPC library and don't make those requests using the loopback interface.

-l

Log security related events to the system log (`syslog`).

-L

Allow old-style local connections over the loopback interface. Without this option, local connections are only allowed over a local socket `/var/run/rpcbind.sock`.

-s

Cause `rpcbind` to change to the user daemon as soon as possible. Using the `rpcbind` utility, you use nonprivileged ports for outgoing connections, and prevent nonprivileged clients to connect to services from a privileged port.

Copyright © 2014, QNX Software Systems Limited

Description:

The `rpcbind` server converts RPC program numbers into universal addresses. You must run this utility on the host to make RPC calls on a server.

When an RPC service is started, it tells `rpcbind` the address at which it is listening and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts `rpcbind` on the server machine to determine the address where RPC requests should be sent.

The `rpcbind` utility should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so `rpcbind` must be started before port monitors are invoked.

When `rpcbind` is started, it checks that certain name-to-address translation calls function correctly. If they fail, the network configuration databases may be corrupt. Since RPC services cannot function correctly in this situation, `rpcbind` reports the condition and terminates.

- The `rpcbind` utility is secure by default, and can be started only by the superuser.
- This utility needs a writable `/var/run` directory in order to run.

Standard RPC servers can be run by `inetd`, so you must start `rpcbind` before `inetd` is invoked.

Files:

The `rpcbind` utility needs `/etc/netconfig`, as well as the following entries in `/etc/services`:

```
sunrpc          111/tcp         rpcbind portmap
sunrpc          111/udp         rpcbind portmap
```

It also requires the `librpc` shared library.

Caveats:

If you restart `rpcbind`, you must restart all RPC servers.

Copyright © 2014, QNX Software Systems Limited

rpcgen

An RPC protocol compiler

Syntax:

```
rpcgen infile

rpcgen -a|-c|-h|-l|-m|-Sc|-Ss [-o outfile] [infile]
       [-C] [-D macro[=value]] [-L] [-K secs]

rpcgen -s transport [-o outfile] [infile]
```

Runs on:

Linux

Options:

-a

Generate *all* the files including sample code for client and server side.

-C

Generate ANSI C code. This option also generates code that could be compiled with the C++ compiler.

-c

Compile into XDR routines.

-D *macro*[=*value*]

Define a macro. It is equivalent to the `#define` directive in the source. If no *value* is given, *value* defaults to 1. This option may be specified more than once.

-h

Compile into C data definitions (a header file).

-K *secs*

Terminate after *secs* seconds of idleness. The default is 120 seconds. -K `0` means exit immediately upon servicing a request (useful if the server is started by `inetd`). -K `-1` disables the idle timer, so the server never exits.

-L

Copyright © 2014, QNX Software Systems Limited

Use *syslog()* for error reporting. In order to capture the log messages, you need to have *syslogd* (p. 1885) running.

-l

("el") Compile into client-side stubs.

-m

Compile into server-side stubs, but don't generate a main routine. You should find this useful for doing callback routines or for when you need to write your own main routine to do initialization.

-o *outfile*

Use this output file. If you don't specify a file, `rpcgen` uses the standard output (-c, -h, -l, -m, -Sc, -Ss, and -s modes only).

-s *transport*

Compile into server-side stubs, using this transport. The compiler supports the transports *udp* and *tcp* (default is *udp*). You can invoke this option more than once to compile a server that serves multiple transports. The *transport*s are chosen at run time and not at compile time.

-Sc

Generate client-side sample code to show the use of remote procedure and how to bind the server before calling the client-side stubs generated by `rpcgen`.

-Ss

Generate skeleton code for the remote procedure on the server side. You need to fill in the actual code for the remote procedures.

infile

Use this input file. If you don't specify an input file in the -c, -h, -l, -m, and -s modes, `rpcgen` uses the standard input.

Description:

The `rpcgen` compiler generates C code to implement an RPC protocol. The input `rpcgen` takes is a C-like language known as Remote Procedure Call Language. For more information about this language, we recommend *Power Programming with RPC* by John Bloomer, O'Reilly & Associates, 1992. ISBN: 0937175773.

Copyright © 2014, QNX Software Systems Limited

You normally use rpcgen in its first form, where it takes an input file and generates four output files. For example, if you specify an *infile* called proto.x, then rpcgen generates:

- a header file in proto.h
- XDR routines in proto_xdr.c
- server-side stubs in proto_svc.c
- client-side stubs in proto_clnt.c

When you use the -Sc option, it generates sample code to illustrate how to use remote procedures on the client side. This code is created in proto_client.c. When you use the -Ss option, it generates sample server code to illustrate how to write remote procedures. This code is created in proto_server.c.

You use the other syntax forms when you want to generate only one of these output files.

Once you create a server, it can be started by the port monitors (for example, inetd or listen) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor 0 was passed. The name of the transport must be specified by setting up the environmental variable PM_TRANSPORT. When the server, generated by rpcgen, is executed, it creates server handles for all the transports specified in the NETPATH environment variable, or if it is unset, it creates server handles for all the visible transports from the /etc/netconfig file.

You use the other syntax forms when you want to generate only one of these output files. When rpcgen is executed with the -s option, it creates servers for that particular class of transports. When executed with the -n option, it creates a server for the transport specified by netid. If infile is not specified, rpcgen accepts the standard input.

The C compiler is used to preprocess all input files before they're actually interpreted by rpcgen, so all the preprocessor directives are legal within an rpcgen input file. For each type of output file, rpcgen defines a special symbol for your use:

rpcgen defines this symbol:	When compiling into:
RPC_HDR	Header files
RPC_XDR	XDR routines
RPC_SVC	Server-side stubs
RPC_CLNT	Client-side stubs

In addition, rpcgen does some preprocessing of its own. It leaves any line beginning with a % uninterpreted and passes the line directly into the output file.

Copyright © 2014, QNX Software Systems Limited

You can customize some of your XDR routines by leaving associated data types undefined. For every data type you leave undefined, `rpcgen` assumes a routine exists whose name starts with *xdr_* and ends with the name of the undefined type.

Caveats:

The compiler doesn't support nesting. You can achieve the same effect, however, by declaring structures at the top level and using their names inside other structures.

When you use program definitions, name clashes can occur since the apparent scoping doesn't apply. You can avoid most of these clashes by assigning unique names to programs, versions, procedures, and types.

The server code generated with the -n option refers to the transport indicated by `netid`, and hence is very site specific.

Copyright © 2014, QNX Software Systems Limited

rpcinfo

Report RPC information

Syntax:

```
rpcinfo -p [host]
rpcinfo [-n portnum] -u host program [version]
rpcinfo [-n portnum] -t host program [version]
rpcinfo -b program version
rpcinfo -d program version
```

Runs on:

QNX Neutrino

Options:

-b *program version*

Using UDP, make an RPC broadcast to procedure 0 of the specified *program* and *version*, and report all hosts that respond.

-d *program version*

Delete the registration for the RPC service of the specified *program* and *version*. Only the superuser can use this option.

-n *portnum*

Use *portnum* as the port number for the -t and -u options instead of the port number given by the portmapper.

-p [*host*]

Probe the portmapper on *host* and print a list of all registered RPC programs. If a host isn't specified, use the standard hostname for the current processor.

-t *host program*

Using TCP, make an RPC call to procedure 0 of *program* on the specified *host* and report whether a response was received.

-u *host program*

Copyright © 2014, QNX Software Systems Limited

Using UDP, make an RPC call to procedure 0 of *program* on the specified *host* and report whether a response was received.

program

A name or a number of a program, as listed by the -p option.

version

A version of the specified program. If you don't specify a version, `rpcinfo` looks for all the registered version numbers for the specified program and then tries to call each registered version. To find all of a program's version numbers, `rpcinfo` calls version 0, which is presumed not to exist. If it does exist, `rpcinfo` tries to obtain this information by calling an extremely high version number instead.

Description:

The `rpcinfo` utility makes an RPC call to an RPC server and reports what it finds.

Examples:

Show all the RPC services registered on the local machine:

```
rpcinfo -p
```

Show all the RPC services registered on the machine named `klaxon`:

```
rpcinfo -p klaxon
```

Delete the registration for version 1 of the `walld` service:

```
rpcinfo -d walld 1
```

Files:

`/etc/rpc`

RPC program number database.

Copyright © 2014, QNX Software Systems Limited

rsh

Remote shell

Syntax:

```
rsh [-dn] [-l username] host [command]
```

Runs on:

QNX Neutrino

Options:

-d

Using *setsockopt()*, turn on socket debugging on the TCP sockets used for communication with the remote host.

host

An Internet address specified in dot notation, or a hostname.

-l *username*

Use the specified remote name (the default remote name is the same as the local username). Authorization is determined as in `rlogin`.

-n

Redirect input from the special device `/dev/null` (see the *"Caveats"* (p. 1704) section below).

command

The command to execute on remote host.

Description:

The `rsh` utility executes *command* on *host*. The standard input of `rsh` is copied to the remote command, and the standard output and standard error of the remote command are copied to `rsh`'s standard output and standard error. Interrupt, quit, and terminate signals are propagated to the remote command; `rsh` normally terminates when the remote command does.

If you don't specify a command, you're logged in on the remote host via `rlogin`.

Copyright © 2014, QNX Software Systems Limited

Shell metacharacters that aren't quoted are interpreted on the local machine; quoted metacharacters are interpreted on the remote machine. For example, the command:

```
rsh otherhost cat remotefile >> localfile
```

appends *remotefile* to *localfile*, while:

```
rsh otherhost cat remotefile ">>" other_remotefile
```

appends *remotefile* to *other_remotefile*.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Files:

`/etc/hosts`

Hostname database.

The `rsh` utility requires the `libsocket.so` shared library.

Caveats:

You can't use `rsh` to run an interactive command such as `vi`; use `rlogin` instead.

Copyright © 2014, QNX Software Systems Limited

rshd

Remote shell daemon

You must be `root` to start this daemon.

Syntax:

```
rshd [-aln]
```

Runs on:

QNX Neutrino

Options:

-a

Request the address for the hostname, and verify whether the address and name correspond.

-l

Unless the user is the superuser, prevent any validation based on the user's `.rhosts` file.

-n

Disable transport-level keepalive messages.

Description:

The `rshd` daemon is the server for the *rcmd()* function and, consequently, for the *rsh* (p. 1703) utility. The daemon provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

The `rshd` daemon is started when *inetd* (p. 977) receives a service request at the port indicated by the *cmd* entry (`inetd` listens for service requests specified in the *inetd.conf* (p. 979) file at a port defined in the *services* (p. 1744) file). Note that the descriptions in the default `inetd.conf` file are commented out; uncomment the ones that you want to use. The following protocol is initiated:

1. The server checks the client's source port. If the port isn't in the range `512-1023`, the server aborts the connection.

Copyright © 2014, QNX Software Systems Limited

2. The server reads characters from the socket up to a `NULL` (`\0`) byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 2 is nonzero, it's interpreted as the port number of a secondary stream to be used for standard error. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range `512-1023`.
4. The server checks the client's source address and requests the corresponding hostname (see *gethostbyaddr()*, the */etc/hosts* (p. 946) file, and *named* (p. 1332)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the -a option is given, the address for the hostname is requested, and the server verifies whether the address and name correspond. If address verification fails, the connection is aborted with the message, "`Host address mismatch`."
5. A `NULL`-terminated username of at most 16 characters is retrieved on the initial socket. This username is interpreted as the user ID on the *client's* machine.
6. A `NULL`-terminated username of at most 16 characters is retrieved on the initial socket. This username is interpreted as a user ID to use on the *server's* machine.
7. A `NULL`-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper boundary on the size of the system's argument list.
8. The `rshd` server then validates the user by means of *ruserok()*, which uses the file `/etc/hosts.equiv` and the file `.rhosts` found in the user's home directory. The -l option prevents *ruserok()* from doing any validation based on the user's `.rhosts` file unless the user is the superuser.
9. A `NULL` byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rshd`.

   Transport-level keepalive messages, which allow sessions to be timed out if the client crashes or becomes unreachable, are enabled unless the -n option is given.

Diagnostics

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of `1` (`0` is returned in step 9 above upon successful completion of all the steps prior to the execution of the login shell).

`Can't fork; try again.`

A *fork()* by the server failed.

`Can't make pipe.`

The pipe needed for standard error wasn't created.

Copyright © 2014, QNX Software Systems Limited

Command too long.

The command line passed is > the size of the argument list (as configured into the system).

Locuser too long.

The name of the user on the client's machine is > 16 characters.

Login incorrect.

No password file entry for the username existed.

Permission denied.

The authentication procedure described above failed.

Remote directory.

The `chdir` command to the home directory failed.

Ruser too long.

The name of the user on the remote machine is > 16 characters.

***<shellname>*: ...**

The user's login shell couldn't be started. This message, which is returned on the connection associated with standard error, isn't preceded by a flag byte.

Caveats:

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. Though insecure, this procedure is useful in an “open” environment.

Copyright © 2014, QNX Software Systems Limited

rtadvd

Router advertisement daemon

Syntax:

```
rtadvd [-c configfile] [-dDfRs] interface ...
```

Runs on:

QNX Neutrino

Options:

-c *configfile*

Specify an alternate location, *configfile*, for the configuration file. By default, `/etc/rtadvd.conf` is used.

-D

Log more verbose debugging information (than the -d option) to *syslogd* (p. 1885).

-d

Log verbose debugging information to *syslogd* (p. 1885).

-f

Prevent `rtadvd` from becoming a daemon (run in foreground mode); this is useful when debugging.

-R

Accept router renumbering requests. If you enable it, certain IPsec setup is suggested for security reasons.

-s

Do not add or delete prefixes dynamically. Only statically configured prefixes, if any, will be advertised.

interface

The interface name(s) to use when sending router advertisement packets.

Description:

The `rtadvd` daemon advertises router advertisement packet to the specified interfaces.

Copyright © 2014, QNX Software Systems Limited

The program will daemonize itself on invocation. It will then periodically send router advertisement packets, as well as in response to router solicitation messages sent by end hosts.

Router advertisements can be configured on a per-interface basis, as described in `rtadvd.conf`.

If there is no configuration file or the interface doesn't have a configuration file entry, `rtadvd` sets all the parameters to their default values. In particular, `rtadvd` reads all the interface routes from the routing table and advertises them as on-link prefixes.

The daemon also watches the routing table. By default, if an interface direct route is added on an advertising interface and no static prefixes are specified by the configuration file, `rtadvd` adds the corresponding prefix to its advertising list. Similarly, if such a route is deleted, `rtadvd` deletes the corresponding prefix from the list. The -s option disables this behavior. Moreover, if the status of an advertising interface changes, `rtadvd` will start or stop sending router advertisements according to the latest status.

Upon receipt of signal `SIGUSR1`, `rtadvd` will dump the current internal state into `/var/run/rtadvd.dump`.

Use SIGTERM to kill `rtadvd` gracefully. In this case, `rtadvd` will transmit router advertisement with router lifetime `0` to all the interfaces (according to RFC2461 6.2.5).

Based on:

Thomas Narten, Erik Nordmark and W. A. Simpson, *Neighbor Discovery for IP version 6 (IPv6)*, *RFC 2461*

Examples:

Start the router advertisement daemon:

```
rtadvd
```

Run the router advertisement program as a foreground process:

```
rtadvd -f
```

Files:

`/etc/rtadvd.conf`

Default configuration file.

`/var/run/rtadvd.pid`

Contains the PID of the currently running `rtadvd`.

`/var/run/rtadvd.dump`

The file in which `rtadvd` dumps its internal state.

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

Success.

> 0

An error occurred.

Contributing author:

Andrey A. Chernov

License:

This utility is based on copyright software of the Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Caveats:

Router advertisements should only be performed downstream. Erroneous upstream advertisements will cause ICMPv6 redirect packet storms in the subnet, as (per the specification) the advertising router is assumed to become the default router for end hosts in the subnet.

Copyright © 2014, QNX Software Systems Limited

/etc/rtadvd.conf

Configuration file for router advertisement daemon

Name:

`/etc/rtadvd.conf`

Description:

This file describes how the router advertisement packet must be constructed for each of the interfaces.

Each line in the file describes a network interface. Fields are separated by a colon (:), and each field contains one capability description. Lines may be concatenated by using the backslash (\) character. The comment marker is the pound sign (#).

Capabilities

Capabilities describe the value to be filled into ICMPv6 router advertisement messages and to control `rtadvd` behavior. Therefore, you are encouraged to read IETF neighbor discovery documents if you would like to modify the sample configuration file.

Almost all items have default values. If you omit an item, the default value of the item will be used.

The following two items control the interval of sending router advertisements:

`maxinterval`

This is the maximum time (in seconds) allowed between sending unsolicited multicast router advertisements. The value must be be no less than 4 seconds and no greater than 1800 seconds. The default value is 600.

`mininterval`

The minimum time (in seconds) allowed between sending unsolicited multicast router advertisements. Its value must be no less than 3 seconds and no greater than .75 * the value of `maxinterval`. The default value is one third of the value of `maxinterval`.

The following items are for ICMPv6 router advertisement message header.

`chlim`

The number value for Cur Hop Limit field. The default value is 64.

`raflags`

Copyright © 2014, QNX Software Systems Limited

A number specifying the Flags field in router advertisement message header. Bit 7 (`0x80`) means Managed address configuration flag bit, and Bit 6 (`0x40`) means Other stateful configuration flag bit. The default value is 0.

`rltime`

The number of seconds specifying the Router lifetime field. Its value must be no greater than 3600000. The default value is 1800.

rtime

The number of milliseconds specifying the Reachable time field. The default value is 0, which means unspecified by this router.

retrans

The number of milliseconds specifying the Retrans Timer field. The default value is 0, which means unspecified by this router.

The following items are for ICMPv6 prefix information option, which will be attached to router advertisement header.

addrs

The number of prefixes. Its default is 0, so it must explicitly be set to positive values if you want to specify any prefix information option. If its value is 0, `rtadvd` looks up the system routing table and advertise the prefixes corresponding to interface routes on the interface. If its value is more than 1, you must specify the index of the prefix for each item below. Indices vary from `0` to `N-1`, where `N` is the value of `addrs`. Each index shall follow the name of each item, e.g. `prefixlen2`.

prefixlen

A number specifying the Prefix length field. The default value is 64.

pinfoflags

A number specifying the Flags field in prefix information option. Bit 7 (`0x80`) means On-link flag bit, and Bit 6 (`0x40`) means Autonomous address-configuration flag bit. The default value is `0xc0`, i.e. both bits are set.

addr

A string specifying the address filled into Prefix field. Since the colon character (`:`) is used for IPv6 numeric address, the field must be quoted by the double-quote character " ". This field cannot be omitted if the value of `addrs` is more than 0.

vltime

The number of seconds specifying the Valid lifetime field. The default value is `2592000` (30 days).

pltime

Copyright © 2014, QNX Software Systems Limited

The number of seconds specifying the Preferred lifetime field. The default value is `604800` (7 days).

The following items are for ICMPv6 MTU option, which will be attached to router advertisement header.

mtu

A number or string that specifies the MTU (maximum transmission unit) field. If `0` is specified, it means that the option will not be included. The default value is `0`. If the special string "auto" is specified for this item, MTU option will be included and its value will be set to the interface MTU automatically.

The following item controls ICMPv6 source link-layer address option, which will be attached to router advertisement header.

nolladdr

A boolean expression. By default (if `nolladdr` is not specified), `rtadvd` will try to get link-layer address for the interface from the kernel, and attach that in source link-layer address option. If this capability exists, `rtadvd`(8) will not attach source link-layer address option to router advertisement packets.

Based on:

Thomas Narten, Erik Nordmark and W. A. Simpson, *Neighbor Discovery for IP version 6 (IPv6)*, *RFC 2461*

Examples:

```
#
# common definitions.
#
default:\
        :raflags#0:rltime#3600:\
        :pinfoflags#64:vltime#360000:pltime#360000:mtu#1500:
ether:\
        :mtu#1280:tc=default:

#
# interfaces.
#
ef0:\
        :addrs#1:\
        :addr="3ffe:501:4819:1000::":tc=ether:
ef1:\
        :addrs#2:addr0="3ffe:501:4819:2000::":\
        :addr1="3ffe:501:4819:3000::":tc=ether:
```

Copyright © 2014, QNX Software Systems Limited

rtc

Set or get date from realtime clock (QNX Neutrino)

You must be `root` to run this utility.

Syntax:

Update the current time based on the time from the specified clock:

```
rtc [-b [base][,[reg_shift][,[mem_map][,c_offset]]]]
    [-l] [-r rate] [-S seconds] clock_type
```

Set the time of the specified clock to the current time:

```
rtc [-b [base][,[reg_shift][,[mem_map][,c_offset]]]]
    -s [-l] clock_type
```

Runs on:

QNX Neutrino

Options:

-b [*base*][,[*reg_shift*][,[*mem_map*][,*c_offset*]]]

The location of the RTC chip:

- *base* — the base address of the chip.
- *reg_shift* — spacing of the device registers as a power of 2. For example:

  0

  Registers are 1 byte apart.

  1

  Registers are 2 bytes apart.

  2

  Registers are 4 bytes apart.

  …

  n

  Registers are 2^n bytes apart.

The default *reg_shift* is `0`.

Copyright © 2014, QNX Software Systems Limited

- *mem_map* — memory or I/O space.
- *c_offset* — offset to the century byte in NVRAM.

-l

("el") Set hardware time to local time, not Coordinated Universal Time (*UTC*). UTC is the standard term for Greenwich Mean Time (*GMT*). The -l option has no effect if the clock_type is `net`.

-r *rate*

Rate at which to adjust the OS time if it's currently within 60 seconds (or value set by -S) of the time from the source specified by *clock_type*. The *rate* is turned into a percentage 1/*rate*. The default rate is 100 (1%).

-S *seconds*

Specify the maximum number of seconds difference between current time and new time before jam loading rather than slowly adjusting. (see -r for the convergence speed). Default: 60.

-s

Set hardware to current date and time.

clock_type

One of the following:

Clock type	Description
hw	Hardware clock (automatically selects one based on information provided by the startup)
at (deprecated)	IBM PC/AT Compatible hardware clock
ds1386	Embedded Dallas Semiconductor DS1386
ps2 (deprecated)	IBM PS/2 Compatible hardware clock
rtc72423	Embedded Fox RTC-72423
mc146818	IBM PC/AT Compatible hardware clock
m48t5x	STMicroelectronics TIMEKEEPER Series clock

Copyright © 2014, QNX Software Systems Limited

Clock type	Description
net [*node*]	Hardware clock on a remote node

Description:

The `rtc` command gets or sets the date and time from a battery backed-up hardware clock.

If your machine has a builtin clock/calendar, you should include the following command in your startup script so QNX Neutrino automatically reads the time when the system starts:

```
rtc hw
```

If the RTC chip has been set to the UTC timezone, startup sets the correct time of day automatically on booting. If the RTC chip is set to local time, or for some reason startup doesn't know where the RTC chip is, you'll need to include the appropriate `rtc` command in the startup script specified in your *`mkifs`* (p. 1241) buildfile.

You can use clock type net [*node*] to get the date from a specified *node*, or to set the date on a specified *node*. If *node* isn't specified, the default is the local machine. When clock type net [*node*] is used, the -l option has no effect.

Be careful if you set the date during the period that a time zone is switching to daylight saving time (DST). When a time zone changes to DST, the local time goes back one hour (for example, 2:00 a.m. becomes 1:00 a.m.). The local time during this hour is ambiguous (e.g. 1:14 a.m. occurs twice in the morning that the time zone switches to DST). To avoid problems, use UTC time to set the date in this period.

Examples:

Update the current date and time from the hardware clock:

```
rtc hw
```

Set hardware clock with current date and time:

```
rtc -s hw
```

Exit status:

0

Successful.

>0

Copyright © 2014, QNX Software Systems Limited

An error occurred.

Copyright © 2014, QNX Software Systems Limited

rtquery

Query routing daemons for their routing tables

Syntax:

```
rtquery [-1np] [-a secret] [-r addr] [-w timeout] host ...
        [-t op] host ...
```

Runs on:

QNX Neutrino

Options:

-1

Query using RIP version 1 instead of RIP version 2.

-a passwd=*XXX* or -a md5_passwd=*XXX*\|*KeyID*

Send the query with the specified RIPv2 cleartext or RIPv2 MD5 password.

-n

Display only the numeric network and host numbers instead of both the numeric and symbolic values.

-p

Use the `poll` command to request full routing information from `gated`. This is an undocumented extension RIP protocol supported only by `gated`.

-r *addr*

Ask about the route to destination *addr*.

-w *timeout*

Change the delay for an answer from each *host*. By default, each *host* has 15 seconds to respond.

-t *op*

Change the tracing settings. Requests from processes that aren't running with `root` privileges, or are on distant networks, are generally ignored by the daemon except for having a message logged in the system log.

on=tracefile

Copyright © 2014, QNX Software Systems Limited

Turn on tracing to the tracefile. The tracefile was either specified when the daemon started, or is the same as a fixed name, typically `/etc/routed.trace`.

more

Increase the debugging level.

off

Turn off tracing.

dump

Dump the daemon's routing table to the current tracefile.

Description:

The `rtquery` utility is used to query a RIP network routing daemon, `routed` or `gated`, for its routing table by sending a `request` or `poll` command. The routing information in any routing response packets returned is displayed numerically and symbolically.

By default, it uses the `request` command. When -p is specified, `rtquery` uses the `poll` command, an undocumented extension to the RIP protocol supported by `gated`. When querying `gated`, the `poll` command is preferred over the `request` command because the response is not subject to *split-horizon* and/or Poisoned Reverse, and because some versions of `gated` do not answer the `request` command. Although `routed` doesn't answer the `poll` command, it recognizes `request`s coming from `rtquery` and answers them completely.

The utility is also used to turn tracing on or off in `routed`.

Based on:

- *RFC 1058* Routing Information Protocol, RIPv1
- *RFC 1723* Routing Information Protocol, RIPv2

An unsupported version of `gated` is available on Foundry27.

Copyright © 2014, QNX Software Systems Limited

rtsold

Router solicitation daemon

Syntax:

```
rtsold [-1Ddfm] -a
rtsold [-1Ddfm] interface ...
rtsol [-Dd] -a
rtsol [-Dd] interface ...
```

Runs on:

QNX Neutrino

Options:

-1

Perform only one probe. Send Router Solicitation packets until valid Router Advertisement packets arrive from all the interfaces, then exit.

-a

Autoprobe the outgoing interface. The `rtsold` daemon tries to find a non-loopback, non-point-to-point, IPv6-capable interface. If it finds multiple interfaces, `rtsold` exits with an error.

-D

Increase the debugging level. Also print internal timer information.

-d

Enable debugging.

-f

Prevent `rtsold` from becoming a daemon (foreground mode). Warning messages are generated to standard error output, instead of *syslog()*.

-m

Enable mobility support; send probing packets to the default routers that advertised Router Advertisements when the node (re)attached to an interface. Periodically send Router Solicitation on an interface that doesn't support `SIOCGIFMEDIA` ioctl.

interface

Copyright © 2014, QNX Software Systems Limited

Name of the interface(s) where messages are to be sent.

Description:

The `rtsold` daemon sends ICMPv6 Router Solicitation messages on the specified *interface*s. If a node (re)attaches to a link, `rtsold` sends some Router Solicitations on the link destined to the link-local scope all-routers multicast address to discover new routers and to get non-link-local addresses.

If you use `rtsol`, probes are transmitted from the *interface* specified, without becoming a daemon. It behaves the same as:

```
rtsold -f1 interface
```

The `rtsold` daemon sends at most three Router Solicitations on an interface after one of the following events:

- Just after the `rtsold` daemon has started.
- The interface is up after a temporary interface failure. Failures are detected by `rtsold` by periodically probing the status of the interface. Some network cards and drivers don't allow the extraction of the link state. In these cases, `rtsold` won't be able to detect a change in the status.
- Every 60 seconds, if -m is specified and `rtsold` can't get the interface status. This feature doesn't conform to IPv6 neighbor discovery specification, but is provided for mobile stations. The default interval for router advertisements, which is on the order of 10 minutes, is slightly long for mobile stations. This feature is provided for such stations so that they can find new routers as soon as possible when they attach to another link.

Once `rtsold` sends a Router Solicitation, and receives a valid Router Advertisement, it refrains from sending additional solicitations on that interface, until the next time one of the above events occurs.

When sending a Router Solicitation on an interface, `rtsold` includes a Source Link-layer address option if the interface has its link-layer address.

When a `SIGUSR1` signal is received, `rtsold` dumps the current internal state into `/var/run/rtsold.dump`.

Files:

`/var/run/rtsold.pid`

The *pid* of the currently running `rtsold`.

`/var/run/rtsold.dump`

Dump of the current internal state.

Copyright © 2014, QNX Software Systems Limited

Exit status:

0

Successful completion.

Nonzero

An error occurred.

Caveats:

Before this utility will work, you need to enable the TCP/IP stack to accept route advertisements, like this:

```
sysctl -w net.inet6.ip6.accept_rtadv=1
```


Copyright © 2014, QNX Software Systems Limited

run-qde

Launch the QNX Momentics Integrated Development Environment

Syntax:

On Windows:

base_directory\run-qde.vbs

On Linux:

base_directory/run-qde.sh

where *base_directory* is where you installed QNX SDP

Runs on:

Linux and Windows

Description:

On Linux and Windows development hosts, `run-qde` sets up your development environment and then launches the QNX Momentics IDE, a powerful set of tools based on the Eclipse Platform. It includes OS-specific plugins that help you create, debug, and examine projects for target systems running the QNX Neutrino RTOS.

For more information, see the IDE *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

ruptime

Show host status of local machines

Syntax:

```
ruptime [-alrtu]
```

Runs on:

QNX Neutrino

Options:

-a

Count all users. Without -a, users who are idle for an hour or more aren't counted.

-l

Sort by load average.

-r

Reverse the sort order.

-t

Sort by uptime.

-u

Sort by number of users.

Description:

The `ruptime` utility gives a status-line-like uptime for each machine on the local network; each uptime is formed from packets that are broadcast once a minute by each host on the network.

Machines for which no status report has been received for 11 minutes are shown as being down.

The default listing is sorted by hostname.

Files:

`/var/rwho/whod.*`

Copyright © 2014, QNX Software Systems Limited

Data files

The `ruptime` utility requires the `libsocket.so` shared library.

Caveats:

QNX Neutrino doesn't calculate load average, so only non-QNX Neutrino hosts can have nonzero load averages.

Copyright © 2014, QNX Software Systems Limited

rwho

Show who's logged in on local machines

Syntax:

```
rwho [-a]
```

Runs on:

QNX Neutrino

Options:

-a

Count all users. Without -a, users who are idle for an hour or more aren't counted.

Description:

The `rwho` utility displays information about the logged-in users, but for all machines on the local network. If no report has been received from a machine for five minutes, `rwho` assumes the machine is down, and doesn't report users last known to be logged into that machine.

If a user hasn't typed to the system for a minute or more, `rwho` reports this idle time. A user who hasn't typed to the system for an hour or more is omitted from the output of `rwho` unless you specify the -a option.

Files:

`/var/rwho/whod.*`

Information about other machines.

The `rwho` utility requires the `libsocket.so` shared library.

Caveats:

This utility is unwieldy when the number of machines on the local net is large. It's unreliable due to protocol issues.

Copyright © 2014, QNX Software Systems Limited

rwhod

System status daemon

You must be `root` to start this daemon.

Syntax:

```
rwhod [-i interval] [-u user]
```

Runs on:

QNX Neutrino

Options:

-i *interval*

Specify the broadcast interval in seconds, or in minutes if *interval* has a suffix of `m`. The default is 3 minutes; the maximum is 11 minutes because higher values will cause *ruptime* (p. 1724) to mark the host as being down.

-u *user*

Drop privileges and become the specified user.

Description:

The `rwhod` daemon is the server that maintains the database used by the `rwho` and `ruptime` utilities. Its operation is based on the network's ability to transmit broadcast messages.

The `rwhod` daemon operates as both a producer and a consumer of status information. As a producer of information, it periodically queries the state of the system and constructs status messages that are broadcast on a network. As a consumer of information, it listens for other `rwhod` daemons' status messages, validating them, then recording them in a collection of files located in the directory `/var/rwho`.

The daemon transmits and receives messages at the port indicated by the `rwho` entry in the */etc/services* (p. 1744) file.

Messages received by the `rwhod` daemon are discarded unless they originated at an `rwhod` daemon's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by `rwhod` are placed in files named `whod.`*hostname* in the directory

Copyright © 2014, QNX Software Systems Limited

/var/rwho. These files contain only the most recent message, in the format described above.

The rwhod daemon generates status messages about every three minutes.

Files:

The rwhod daemon requires the libsocket.so shared library.

Copyright © 2014, QNX Software Systems Limited

Chapter 20
S

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following have been deprecated:

Instead of:	Use:
`sin`	*`pidin`* (p. 1521)

The following are described elsewhere:

For information about:	See:
`screeninfo`	*Screen Graphics Subsystem Developer's Guide*
`screen`	*Screen Graphics Subsystem Developer's Guide*

This chapter describes the utilities, etc. whose names start with “S”.

Copyright © 2014, QNX Software Systems Limited

scp

Secure copy (remote file copy program)

Syntax:

```
scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
    [-l limit] [-o ssh_option] [-P port] [-S program]
    [[user@]host1:]file1 ... [[user@]host2:]file2
```

Runs on:

QNX Neutrino

Options:

See *scp* in the NetBSD documentation.

Description:

The scp utility copies files between hosts on a network. It uses ssh for data transfer, and uses the same authentication and provides the same security as ssh. For more information, see *scp* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

script

Create a typescript of a terminal session

Syntax:

```
script [-a] [file]
```

Runs on:

QNX Neutrino

Options:

-a

Append the output to *file* or to the `typescript` file, retaining the prior contents.

file

Name of the file `script` will use to save the session output. If you don't specify a file, output is saved in the file `typescript`.

Description:

The `script` utility records everything printed to the terminal during a session. This is useful, for example, if you want to record an interactive session and then save the `typescript` file for later analysis, tech support, etc.

The script ends when the forked shell exits (via **Ctrl**–**D**).

Certain interactive utilities (e.g. `vi`) can create garbage in the `typescript` file. The `script` command works best with utilities that don't manipulate the screen; the results are meant to emulate a hardcopy terminal.

Note also that `script` records *all* transactions, including linefeeds and backspaces.

Copyright © 2014, QNX Software Systems Limited

sed

Stream editor (POSIX)

Syntax:

```
sed [options]... {script-only-if-no-other-script}
[input-file]...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-e *script* or --expression=*script*

Use the command-line *script* for editing *input-file*s. If you specify multiple -e options, the *script*s are applied in the order specified to each line of the input files. If a -f option is specified in addition to -e, lines are acted upon by *script*s first.

-f *script_file* or --file=*script-file*

Use the file *script_file* as the script of editing instructions. If multiple -f options are specified, the *scripts* are applied in the order specified to each line of the input files. If a -e option is specified in addition to -f, lines are acted upon by *script*s first.

--help

Display some help, and then exit.

-i[*suffix*] or --in-place[=*suffix*]

Edit files in place (making a backup if you specify the *suffix*).

-l *N* or --line-length=*N*

("el") Specify the desired line-wrap length for the `l` command.

-n or --quiet or --silent

Suppress the default output, which passes each line to standard output after it's examined for editing. Only lines explicitly selected for output are written.

--posix

Disable all GNU extensions.

Copyright © 2014, QNX Software Systems Limited

-r or --regexp-extended

Use extended regular expressions in the script.

-s or --separate

Consider files as separate rather than as a single continuous long stream.

-u or --unbuffered

Load minimal amounts of data from the input files and flush the output buffers more often.

--version

Display the version number, and then exit.

If you don't specify any -e, --expression, -f, or --file options, then the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if you don't specify any input files, then the standard input is read.

Description:

The sed utility is a stream editor that reads one or more text files, makes editing changes according to editing commands that you specify, and writes the results to standard output.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

The sed commands are usually stored in a program file (*script_file*), although you may give simple sed commands from the command line. By default, sed copies files from the file list to its standard output, editing the lines in the process. Conceptually, there is *one* input file, which is the concatenation of all input files specified.

Lines are selected for editing based on their position within the input file, or by pattern matching. If no files are listed, input is taken from standard input (this is the only time standard input is used). The sed utility initially reads all the editing commands from all specified sources and places them in an internal table in the order specified. The utility then processes the (concatenated) input file as follows:

1. Read one line from the input file and copy to the *pattern space* (a work area).
2. For each command that selects the line, act on the pattern space as the edit command specifies, cyclically placing the result into the pattern space.
3. After all commands have been checked against the pattern space, write the pattern space to the standard output, provided -n was *not* specified. The pattern space may contain zero, one, or several lines at this time.
4. Delete the contents of the pattern space.

Copyright © 2014, QNX Software Systems Limited

5. Repeat from step 1 until all of the (concatenated) input file has been read and processed.

Scripts

A *script* consists of editing commands, one per line, of the following form:

```
[address[,address]]function[arguments]
```

You can specify the *script* of editing commands in the command line, or it can be contained in the file *script_file*.

In default operation, sed cyclically copies a line of input into a pattern space (unless there is something left after a D command), applies in sequence all commands whose addresses select that pattern space, and at the end of the script copies the pattern space to the standard output (except under -n) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval. The *pattern* and *hold spaces* are each limited to 20 KB.

Addresses

An address is one of the following:

- a decimal number that counts input lines cumulatively across files
- a $ token that addresses the last line of input
- a context address
- a regular expression

A command line with no address selects every pattern space. A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second address. (If the line number of the second address is less than or equal to the line number first selected, then only the first line is selected.) Starting at the first line following the selected range, sed looks again for the first address. Thereafter the process is repeated.

You can use the negation character (!) to apply editing commands to non-selected pattern spaces. For more information, see "*Editing commands* (p. 1735)", below.

Regular expressions

The sed utility uses basic regular expressions (*RE*s), with the following additions:

- In a context address, the construction *?RE?* (where *?* is any character) is mapped to /*RE*/. Note that in the context address \xabc\xdefx, the second x stands for itself, so that the regular expression is abcxdef.
- The escape sequence \n matches a newline embedded in the pattern space.
- A period (.) matches any character except the last newline of the pattern space.

Copyright © 2014, QNX Software Systems Limited

Editing commands

In the following list of functions, the maximum number of permissible addresses for each function is indicated by one of [*0addr*], [*1addr*] or [*2addr*] representing a maximum of zero addresses, one address or two addresses respectively. The argument *text* consists of one or more lines. Each embedded `newline` in the text must be preceded by a backslash. Other backslashes in text are treated like the backslashes in the replacement string of an `s` editing command; you can use them to protect initial `blank`s against the stripping that's done on every script line.

The `r` and `w` editing commands take an optional *rfile* (or *wfile*) parameter, separated from the command letter by zero or more `blank`s.

The arguments *rfile* or *wfile* terminate the command line. Each *wfile* is created before processing begins. There can be at most ten distinct *wfile* arguments in the script.

The `b`, `r`, `s`,`t`, `w`, `y`, `!`, and `:` editing commands take additional arguments. The following synopses indicate which arguments are separated from the commands by blanks:

[*2addr*] { *command_list* or }

Execute *command_list* only when the pattern space is selected. (Note that the trailing `}` must be the first non-blank character in the line.)

[*1addr*]a\ or *text*

Write *text* to the standard output after the pattern space is written.

[*2addr*]b *label*

Branch to the `:` (colon) command bearing the *label*. If *label* is empty, branch to the end of the script.

[*2addr*]c\ or *text*

Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output.

[*2addr*]d

Delete the pattern space and start the next cycle.

[*2addr*]D

Delete the initial segment of the pattern space through the first `newline` and start the next cycle.

[*2addr*]g

Replace the contents of the pattern space by the contents of the hold space.

Copyright © 2014, QNX Software Systems Limited

[*2addr*]G

Append the contents of the hold space to the pattern space.

[*2addr*]h

Replace the contents of the hold space by the contents of the pattern space.

[*2addr*]H

Append the contents of the pattern space to the hold space.

[*1addr*]i\ or *text*

Write *text* to the standard output before the pattern space is written.

[*2addr*]l

List the pattern space on the standard output in an unambiguous form. Nonprinting characters are listed as hexadecimal digit pairs, with a preceding backslash, with the following exceptions:

Character	Listed as
alert	\a
backslash	\\
backspace	\b
carriage return	\r
form-feed	\f
newline	\n
tab	\t
vertical tab	\v

Long lines are folded; the length at which folding occurs is unspecified, but should be appropriate for the output device.

[*2addr*]n

Copy the pattern space to the standard output and replace the pattern space with the next line of input.

[*2addr*]N

Append the next line of input to the pattern space, using an embedded newline to separate the appended material from the original material. Note that the current line number changes.

Copyright © 2014, QNX Software Systems Limited

[*2addr*]p

Copy (print) the pattern space to the standard output.

[*2addr*]P

Copy (print) the pattern space, up to the first `newline`, to the standard output.

[*1addr*]q

Branch to the end of the script and quit without starting a new cycle.

[*1addr*]r *rfile*

Read the contents of the *rfile* file. The contents are placed on the output before reading the next input line.

[*2addr*]s*/ regular expression/ replacement string/ flags*

Substitute the *replacement string* for instances of *regular expression* in the pattern space. You can use any character instead of `/`. The value of *flags* is zero or more of:

n

n=1 to 512. Substitute for the *n*th occurrence only of the *regular expression* found within the pattern space.

g

Globally substitute for all non-overlapping instances of the *regular expression* rather than just the first one. If both `g` and `n` are specified, `g` takes precedence.

p

Print the pattern space if a replacement was made.

w *wfile*

Append (write) the pattern space to *wfile* if a replacement was made.

[*2addr*]t *label*

Test; branch to the `:` (colon) command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a `t`. If *label* is empty, branch to the end of the script.

[*2addr*]w *wfile*

Copyright © 2014, QNX Software Systems Limited

Append (write) the pattern space to *wfile*.

[*2addr*]x

Exchange the contents of the pattern and hold spaces.

[*2addr*]y*/string1/string2/*

Replace all occurrences of collating elements in *string1* with the corresponding collating element in *string2*. The lengths of *string1* and *string2* should be equal.

[*2addr*]! *function*

Apply the *function* (or command list, if *function* is `{`) only to the lines that aren't selected by the addresses.

[*0addr*]: *label*

This command does nothing; it bears a *label* for the `b` and `t` commands to branch to.

[*1addr*]=

Place the current line number on the standard output as a line with its own line number.

[*0addr*]

An empty command; ignored.

[*0addr*]#

If a # appears as the first character on any line of a script file, that entire line is ignored (treated as a comment), with the single exception that if the first line of the script file begins with #`n`, the default output is suppressed (as when the -n option is specified on the command line).

For more information, see:

- Dale Dougherty, *sed & awk*, O'Reilly and Associates, 1990.
- Brian W. Kernighan and Rob Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984.

Examples:

In the file `myfile`, find and output only those lines containing the string "`tom`":

```
sed -n -e "/tom/p" myfile
```


Copyright © 2014, QNX Software Systems Limited

In the file `myfile`, replace all occurrences of the string `beneath` with the string `below`, and output to the file `newfile`:

```
sed -e "s/beneath/below/" myfile >newfile
```

Files:

All files are text files. The *script_files* named by the -f option consist of editing commands, one per line. Any number of additional text input files may be specified by the `r` command for insertion of *unedited* data to the standard output at points predetermined by editing rules. A maximum of 10 additional output files may be specified through the use of the `w` command in the script.

Exit status:

0

Successful completion.

>0

An error occurred.

Errors:

If one or more of the input files (this doesn't include script files) can't be opened for reading, `sed` continues to process the remaining files.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

seedres

Seed system resources on x86 platforms

Syntax:

```
seedres
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

None.

Description:

The `seedres` utility reads the PnP BIOS and assigns resources, such as I/O ports and IRQs, to *procnto* (p. 1586)'s resource database. You need it if you want drivers to assign nonconflicting resources.

This utility is for x86 platforms; on other platforms, the startup code seeds the system resources.

For more information about the resource database, see *rsrcdbmr_attach()* in the QNX Neutrino *C Library Reference*.

Examples:

The following lines from a buildfile show how to start `seedres` and *pci-bios* (p. 1455) on an x86 platform:

```
...
seedres
pci-bios
...
```


Copyright © 2014, QNX Software Systems Limited

sendnto

Send an OS image to a target over a serial or parallel port (QNX 4)

Syntax:

```
sendnto -d device [-b baud] [-l speed] [-eqv] filename
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-b *baud*

Set the serial port transfer speed to *baud*.

-d *device*

Send the image over this device. You must specify this option. The form of the device name depends on the host OS:

- Linux: `/dev/ttyS`*X*
- QNX Neutrino: `/dev/ser`*X*
- Windows: `com`*X*

-e

Request an echo from the target for every data record (off by default). An echo may be needed for serial downloads at very high baud rates (>57600 baud) to very slow machines.

-l *speed*

("el") Output is to a parallel port with a LAPLINK cable. The *speed* may be 1...3, with 1 being the fastest.

-q

Be quiet; don't print an ongoing percentage of image downloaded.

-v

Be verbose; print an ongoing percentage of download complete.

Copyright © 2014, QNX Software Systems Limited

Description:

The sendnto utility takes an OS image built with *mkifs* (p. 1241) and sends it to a target device using a fast binary protocol. The target computer needs a loader built into its startup code in ROM or FLASH that understands this protocol. The protocol is very simple and is designed to be implemented with minimal code in the target.

The Board Support Packages generally contain IPL source for serial targets.

The sendnto utility assumes the target has been reset and is in a state where it is waiting for data. It sends the following sequence of records:

```
START
DATA DATA ...
GO
```

Each record has a sequence number and checksum to ensure data integrity. The GO record transfers control to the downloaded image. The details of this protocol may be determined by examining the source to sendnto, which is also made available for porting to other development environments.

Downloads over a parallel port are automatically flow-controlled. Downloads over a serial port assume that the target is fast enough to process the data. For most targets this is true for baud rates up to 57600 baud.

On slower targets with a baud rate of 115200, you may need to specify the -e option to flow control sendnto with the target. This causes sendnto to insert an ECHO record after each data record. It then pauses and waits for the target to echo of a + before sending the next record. If the target fails to send a + within 1/10 second, sendnto times out and sends the next record anyway. If you don't specify the -e option, the link need only operate in one direction (write without read from the host's point of view).

When used over a serial link, sendnto uses the existing baud rate and hardware flow control set by the stty command. It programs the link to operate in raw mode. Note that hardware flow control isn't required on the link.

The -v option makes sendnto print a continuously updated percentage as the image is downloaded. This provides feedback during large image downloads over slow links.

Examples:

Send the file image to the target machine through the first serial port on host machine:

```
sendnto -d /dev/ser1 image
```

Send the file image to the target machine through the second serial port on the node named server:

```
sendnto -d /net/server/dev/ser2 image
```


Copyright © 2014, QNX Software Systems Limited

Send the file `image` to the target machine through the parallel port on the host machine using a regular parallel cable:

```
sendnto -d /dev/par image
```

Send the file `image` to the target machine through the parallel port using a LAP-LINK cable. This cable arrangement may also be used by the `wd` source-level debugger for parallel-port debugging:

```
sendnto -d /dev/bipar image
```

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

/etc/services

Service name database (UNIX)

Name:

`/etc/services`

Description:

The `/etc/services` file contains information regarding the known services available in the DARPA Internet. For each service, a single line should be present with the following information:

```
official_service_name  port_number/protocol_name  aliases
```

For example:

```
ftp             21/tcp
```

The fields are as follows:

ftp

Service name.

21

Port number.

tcp

Protocol name.

Items are separated by any number of blanks, tabs, or both.

The *port_number* and *protocol_name* are considered a single item; a `/` is used to separate the two (e.g. `512/tcp`).

A # in a line indicates the beginning of a comment — any characters after a #, up to the end of the line, aren't interpreted by the routines that search the file.

Service names may contain any printable character other than a field delimiter, newline, or comment character.

Copyright © 2014, QNX Software Systems Limited

setconf

Set a configuration string (QNX)

You must be logged in as `root` to use this utility.

Syntax:

```
setconf variable_name string ...
```

Runs on:

QNX Neutrino

Options:

variable_name

The variable to set; see <pathconf.h>. The most useful variables are:

- _CS_ARCHITECTURE
- _CS_DOMAIN
- _CS_HOSTNAME

A hostname can consist only of letters, numbers, and hyphens, and must not start or end with a hyphen. For more information, see *RFC 952*.

If you change this configuration string, be sure you also change the ***HOSTNAME*** environment variable. The *hostname* (p. 945) utility always gives the value of the _CS_HOSTNAME configuration string.

- _CS_HW_PROVIDER
- _CS_HW_SERIAL
- _CS_LIBPATH
- _CS_LOCALE
- _CS_MACHINE
- _CS_PATH
- _CS_RELEASE
- _CS_RESOLVE
- _CS_SRPC_DOMAIN

Copyright © 2014, QNX Software Systems Limited

- _CS_SYSNAME
- _CS_TIMEZONE
- _CS_VERSION

string

The new value for the variable.

Description:

This utility sets a variable that's used for configuration information. For more information, see the documentation for *getconf* (p. 897).

Copyright © 2014, QNX Software Systems Limited

setfacl

Set the access control list (ACL) for files or directories

Syntax:

```
setfacl [-bn] [-m entry[,entry...]] [-M acl_file]
        [-x entry[,entry...]] [-X acl_file] [path ...]
```

Runs on:

QNX Neutrino

Options:

The options are processed in the order they appear on the command line.

-b

Remove all the extended ACLs from the specified files and directories.

-M *acl_file*

Merge the entries (see below) given in *acl_file* into the ACLs of the specified files and directories. If you specify a hyphen (`-`) for *acl_file*, `setfacl` reads the entries, one per line, from standard input until you press **Ctrl**–**D**.

-m *entry*[,*entry*...]

Merge the given entries into the ACLs of the specified files and directories.

-n

Don't recalculate the permissions associated with the ACL mask entry.

-X *acl_file*

Remove the entries given in *acl_file* from the ACLs of the specified files and directories. If you specify a hyphen (`-`) for *acl_file*, `setfacl` reads the entries, one per line, from standard input until you press **Ctrl**–**D**.

-x *entry*[,*entry*...]

Remove the given entries from the ACL of the specified files and directories.

***path* ...**

Copyright © 2014, QNX Software Systems Limited

The file or directory that you want to set the ACL for. If you specify a hyphen (`-`) or don't specify any paths, `setfacl` reads them, one per line, from standard input until you press **Ctrl**–**D**.

Description:

The `setfacl` utility modifies the access control list for files or directories. ACLs extend the traditional permissions as set with *chmod* (p. 124), giving you finer control over who has access to what. The classes of permissions are:

- owner class
- group class, consisting of named users, the owning group, and named groups
- others (or world) class

For an overview of ACLs, see "Access Control Lists (ACLs)" in the QNX Neutrino *User's Guide*.

If you're using the -M or -m option, each entry is in one of the following forms:

Entry type	Form
Owner	`user::`*permissions*
Named user (identified by name or by ID)	`user:`*user_name*`:`*permissions*
Owning group	`group::`*permissions*
Named group (identified by name or ID)	`group:`*group_name*`:`*permissions*
The upper bound on permissions for the group class.	`mask::`*permissions*
Others	`other::`*permissions*

If you wish, you can specify just the first letter of `user`, `group`, `mask`, and `other`. The permissions are a combination of `r` (read), `w` (write), `x` (execute), and `-` (no permission). You can put these characters in any order and omit any "no permission" hyphens. If the entry already exists in the ACL, the existing permissions are replaced by the new ones.

If you're using the -X or -x option, the permissions are ignored, and *all* permissions are removed for the specified mask, named user, or named group. You can't remove the entries for the owner, owning group, or others.

- Changes that you make to ACLs are kept only until you reboot the system.
- Changes to an ACL can affect the file permissions (i.e., permissions as could be set by `chmod`). The changes to the file permissions do persist across reboots.

Copyright © 2014, QNX Software Systems Limited

Examples:

Add read-only permission for a specific user:

```
# setfacl -m user:frank:r my_file
# getfacl my_file
# file: my_file
# owner: mabel
# group: docs
user::rw-
user:frank:r--
group::rw-
mask::rw-
other::r--
```

Remove the permissions specified in `my_acl` from a file:

```
# cat my_acl
user:frank:
# setfacl -X my_acl my_file
```

Exit status:

0

Success.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

setkey

Manually manipulate the IPsec SA/SP database

Syntax:

```
setkey [-knrv] filename
setkey [-v] [-c]
setkey [-krv] [-f] filename
setkey [-aklPrv] -D
setkey [-Pvp] -F
setkey [-H] -x
setkey [-?V]
```

Runs on:

QNX Neutrino

Options:

-a

Display dead SAD (Security Association Database) entries. A SAD entry is dead when it has expired, but it may still be referenced by SPD (Security Policy Database) entries.

-c

Specify an operation from standard input. For a list of valid operations, see the "*Operations* (p. 1751)" section, below.

-D

Dump the SAD entries.

When used with:	Also dump:
-a	Dead entries
-P	SPD entries

-F

Flush the SAD entries. When specified with -P, also flush the SPD entries.

-f *filename*

A file that contains the operations to perform. For more information, see the "*Operations* (p. 1751)" section, below.

Copyright © 2014, QNX Software Systems Limited

-h

Dump entries in a hexadecimal format.

-l

Loop forever with short output on -D.

-P

Dump (when specified with -D) or flush (with -F) the SPD entries.

-v

Be verbose. Display messages transmitted to the PF_KEY socket (including messages sent from other processes).

-x

Loop forever and dump all the messages transmitted to the PF_KEY socket.

Description:

The `setkey` utility adds, updates, dumps, or flushes the Security Association Database (SAD) entries and the Security Policy Database (SPD) entries in the stack.

Operations

The following operations may be specified from either standard input (using -c) or from a file (using -f *filename*).

Lines that start with hash marks (#) are treated as comment lines. Operations have the following grammar:

add ***src dst protocol spi*** **[*extensions*]** ***algorithm...*** **;**

Add an SAD entry. This operation can fail, for example, if the key length doesn't match the specified algorithm.

delete ***src dst protocol spi*** **;**

Remove an SAD entry.

dump [*protocol*] ;

Dump all SAD entries matched by this *protocol* (same functionality as -D on the command line).

flush [*protocol*] ;

Clear all SAD entries matched by this *protocol* (same functionality as -F on the command line).

Copyright © 2014, QNX Software Systems Limited

get ***src dst protocol spi*** **;**

Show an SAD entry.

spdadd ***src_range dst_range upperspec policy*** **;**

Add an SPD entry.

spddelete ***src_range dst_range upperspec*** **-P** ***direction*** **;**

Delete an SPD entry.

spddump ;

Dump all SPD entries (same functionality as -DP on the command line).

spdflush ;

Clear all SPD entries (same functionality as -FP on the command line).

Meta-arguments for operations

The meta-arguments for the operations are as follows:

algorithm

Specify an encryption, authentication, or compression algorithm.

See the "*Algorithms for protocol* (p. 1755)" section below for a list of the valid values for *aalgo*, *ealgo* and *calgo*.

-A ***aalgo key***

Specify an authentication algorithm (*aalgo*) for the ah and ah-old protocols.

-E ***ealgo key***

Specify an encryption algorithm (*ealgo*) for the esp or esp-old protocols.

-E ***ealgo key*** **-A** ***aalgo key***

Specify an encryption algorithm (*ealgo*) for the esp or esp-old protocols, as well as a payload authentication algorithm (*aalgo*) for esp.

-C ***calgo*** **[-R]**

Specify a compression algorithm for IPComp (IP Payload Compression Protocol).

Copyright © 2014, QNX Software Systems Limited

If -R is specified, the value of the *spi* field is used as the IPComp CPI (compression parameter index) field on outgoing packets. The field must be smaller than `0x10000`.

If -R isn't specified, the stack uses the IPComp CPI (compression parameter index) from the IPComp CPI field on the packets, and the *spi* field is ignored.

key

A double-quoted character string or series of hexadecimal digits preceded by `0x`.

dst*, *src

Specify the destination or source of the secure communication as an IPv4/v6 address. The address must be in numeric form since `setkey` doesn't consult hostname-to-address for these arguments.

dst_range*, *src_range

Selections of the secure communication specified as an IPv4/v6 address or an IPv4/v6 address range. They may accompany TCP/UDP port specifications. Valid forms are:

```
address
address/prefixlen
address[port]
address/prefixlen[port]
```

The values for *prefixlen* and *port* must be specified as a decimal number; *src* and *dst* must be expressed in numeric form. The square brackets around *port* are part of the syntax; they aren't optional.

extensions

Valid options are:

-f *nocyclic-seq*

Don't allow cyclic sequence numbers.

-f *pad_option*

Specify the content of the esp padding, where *pad_option* is one of:

- random-pad — set a series of randomized values.

Copyright © 2014, QNX Software Systems Limited

- seq-pad — set a series of sequential increasing numbers starting from 1.
- zero-pad — set everything to zero.

-lh ***time***

Specify a hard lifetime.

-ls ***time***

Specify a soft lifetime.

-m ***mode***

Security protocol mode to be used, which is one of:

- any (the default) — use whichever security protocol mode is available.
- transport — protect peer-to-peer communication between end nodes.
- tunnel — include IP-in-IP encapsulation operations. It's designed for security gateways like VPN configurations.

-r ***size***

The window size, in bytes, for replay prevention. The value of *size* is a decimal number in a 32-bit word. If *size* is zero or not specified, replay check doesn't take place.

-u ***id***

Specify the identifier in order to relate the *policy* with the SA. The value of *id* must be a decimal number between 1 and 32767.

policy

Takes the form:

```
-P direction discard
-P direction ipsec request ...
-P direction none
```

See “Setting the policy” in the IPsec protocols page for detailed descriptions of the above arguments.

protocol

Valid options are:

Copyright © 2014, QNX Software Systems Limited

- ah — AH (Authentication Header) based on *RFC 2402*.
- ah-old — AH based on *RFC 1826*.
- esp — ESP (Encapsulated Security Payload) based on *RFC 2405*.
- esp-old — ESP based on *RFC 1827*.
- ipcomp — IPCOMP (IP Payload Compression Protocol).

spi

Security Parameter Index (SPI) for the SAD and the SPD. It's a decimal number, or a hexadecimal number prefixed with `0x`. SPI values between the range 0 and 255 are reserved for future use.

upperspec

Specify the upper-layer *protocol* to use, which is one of:

- any (use any protocol)
- tcp
- udp.

Currently, *upperspec* doesn't work against forwarding.

Algorithms for *protocol*

The following tables show the algorithm to use for each *protocol* parameter. The *protocol* and *algorithm* parameters are almost orthogonal.

Authentication algorithms for *aalgo* include:

Algorithm:	Keylen (bits):	Comment:
`hmac-md5`	128	ah: *RFC 2403*; ah-old: *RFC 2085*
`hmac-sha1`	160	ah: *RFC 2404*; ah-old: 128-bit ICV (no document)
`hmac-sha256`	256	ah: 96-bit ICV (draft-ietf-ipsec-ciph-sha-256-00); ah-old: 128-bit ICV (no document)
`hmac-sha384`	384	ah: 96-bit ICV (no document); ah-old: 128-bit ICV (no document)

Copyright © 2014, QNX Software Systems Limited

Algorithm:	Keylen (bits):	Comment:
`hmac-sha512`	512	ah: 96-bit ICV (no document); ah-old: 128-bit ICV (no document)
`hmac-ripemd160`	160	ah: 96-bit ICV (*RFC 2857*); ah-old: 128-bit ICV (no document)

Encryption algorithms for *ealgo* include:

Algorithm	Keylen (bits)	Comment
`des-cbc`	64	esp-old: *RFC 1829*, esp: *RFC 2405*
`3des-cbc`	192	*RFC 2451*
`blowfish-cbc`	40 to 448	*RFC 2451*
`cast128-cbc`	40 to 128	*RFC 2451*
`des-32iv`	64	esp-old: RFC 1829
`des-deriv`	64	*ipsec-ciph-des-derived-01* (expired)
`3des-deriv`	192	No document
`rijndael-cbc`	128/192/256	*draft-ietf-ipsec-ciph-aes-cbc-00*

Compression algorithms for *calgo* include:

Algorithm	Comment
`deflate`	*RFC 2394*

Examples:

```
add  3ffe:501:4819::1 3ffe:501:481d::1 esp 123457
        -E des-cbc "ESP SA!!" ;

add 3ffe:501:4819::1 3ffe:501:481d::1 ah 123456
        -A hmac-sha1 "AH SA configuration!" ;

add 10.0.11.41 10.0.11.33 esp 0x10001
        -E des-cbc "ESP with"
        -A hmac-md5 "authentication!!" ;

get 3ffe:501:4819::1 3ffe:501:481d::1 ah 123456 ;

flush ;

dump esp ;

spdadd  10.0.11.41/32[21] 10.0.11.33/32[any] any
        -P out ipsec esp/tunnel/192.168.0.1-192.168.1.2/require ;
```


Copyright © 2014, QNX Software Systems Limited

Exit status:

0

Success.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

sftp

Secure file transfer program

Syntax:

```
sftp [-1Cv] [-B buffer_size] [-b batchfile] [-F ssh_config]
     [-o ssh_option] [-P sftp_server_path] [-R num_requests]
     [-S program] [-s subsystem | sftp_server] host

sftp [[user@]host[:file [file]]]

sftp [[user@]host[:dir[/]]]

sftp -b batchfile [user@]host
```

Runs on:

QNX Neutrino

Options:

See *sftp* in the NetBSD documentation.

Description:

The `sftp` utility is an interactive file transfer program, similar to *ftp* (p. 848), which performs all operations over an encrypted `ssh` transport. For more information, see *sftp* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

sftp-server

SFTP server subsystem

Syntax:

```
sftp-server [-f log_facility] [-l log_level]
```

Runs on:

QNX Neutrino

Options:

See *sftp-server* in the NetBSD documentation.

Description:

The `sftp-server` is a program that speaks the server side of the SFTP protocol to *stdout* and expects client requests from *stdin*. The `sftp-server` program isn't intended to be called directly, but from `sshd` using the Subsystem option.

For more information, see *sftp-server* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

sh

Public domain Korn shell command interpreter (UNIX)

Syntax:

```
sh [+-abCefhikmnprsuvxX] [+-o option]
   [ [ -c command-string [command-name]
     | -s | file ] [argument ...] ]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

See *ksh* (p. 1029).

Description:

The sh shell is a public-domain version of the Korn shell. For more information, see *ksh* (p. 1029).

Copyright © 2014, QNX Software Systems Limited

showlicense

Display the type of QNX license that's currently active (QNX Neutrino)

Syntax:

```
showlicense
```

Runs on:

Linux, Windows

Options:

None.

Description:

The `showlicense` displays the type (e.g., Commercial, Evaluation) of the active QNX Neutrino license. If possible, it also displays a path to the appropriate license text.

Copyright © 2014, QNX Software Systems Limited

showmem

Display memory information

Syntax:

```
showmem [-D type] [-d file] [-P [pid]] [-S] [-v[v]...]
```

Runs on:

QNX Neutrino

Options:

-D *type*

Show detailed process information. The *type* is one or more of the following letters:

- l — libraries; the contribution from the shared objects
- s — stack; the contribution from the threads
- h — heap (not currently implemented)

-d *file*

Dump the raw mapinfo into the specified file.

-P [*pid*]

Show a summary of the memory usage for the processes and the shared libraries in the system. If you specify *pid*, showmem displays information only for the process with that ID.

-S

Show a summary of the free and used memory for the entire system.

-v[v]...

Increase the verbosity of the output.

You must specify at least one of these options.

Description:

The showmem utility displays memory information to help you determine how your system is using memory.

-S option

Copyright © 2014, QNX Software Systems Limited

If you specify the -S option, the output includes a summary for the whole system. For example:

```
System RAM:      511M (     536394752)
Total Used:      381M (     399794216)
 Used Private:   378M (     396812328)
 Used Shared:   2912K (       2981888)
```

The fields are as follows:

- System RAM: total RAM of the system
- Total used: the sum of the Used Private and Used Shared
- Used Private: private blocks, plus shared objects that are referenced by only one process
- Used Shared: shared blocks that are referenced by more than one process

-P option

Use the -P option to display a summary of memory usage by the processes and shared libraries in the system. If you specify a *pid*, `showmem` displays the memory information of the specified process; if you don't, `showmem` displays the memory summary of all processes and shared objects.

The report is broken down into these sections:

Process

The Process section shows the memory that the process claims it's consuming. The fields are:

- Total: sum of Code + Data + Heap + Stack + Other
- Code: the code/text section of the process + the code/text section of the shared libraries being used only by that process
- Data: the data/bss section of the process + the data/bss section of the shared libraries referenced by that process
- Heap: the heap of the process
- Stack: the stack of the process
- Other: other memory not falling into the types above, mainly shared memories referenced by the process
- process ID
- the name of the process

Let's take the process `devc-con-hid` as an example:

```
Process listing (Total, Code, Data, Heap, Stack, Other)
   ...
819200  602112   24576  167936   24576        0    4103  devc-con-hid
```

Total = 602112 + 24576 + 167936 + 24576 + 0 = 819200.

Shared

Copyright © 2014, QNX Software Systems Limited

The Shared section shows all of the objects that have been identified as being shared by more than one process mapping. The fields are:

- Total: sum of Code + Data + Heap + Stack + Other
- Code: the code/text section of the shared library
- Data: 0
- Heap: 0
- Stack: 0
- Other: other memory not falling into the types above, mainly shared memory objects
- pid: -1
- name of the shared object

For example, `libc.so.3` is referenced by almost all the processes. Therefore, it's shown in the Shared section; the information for it could be like this:

```
Shared shared objects (Total, Code, Data, Heap, Stack, Other)
    ...
499712  499712     0     0     0     0     -1  proc/boot/libc.so.3
```

Here's another example of an entry for shared memory in the Shared section:

```
Shared shared objects (Total, Code, Data, Heap, Stack, Other)
     ...
4096     0     0     0     0  4096    -1  /dev/shmem/ctl-1002,5964,0
```

-D option

If you specify the -D option, `showmem` shows detailed process information. The Process section shows the breakdown of the code, data, stack, heap, and libraries being used by the processes. The Shared section is just the same as the Shared section shown with the -P option.

If a value is shown in parentheses, this memory has already been accounted for the total memory information. This memory mapping might originate from within another, not allocated from system RAM.

For example, if you specify -Dsl, the information for `devc-con-hid` might look like this:

```
Process listing (Total, Code, Data, Heap, Stack, Other)
   ...
   319488    102400     24576    167936     24576         0       4103 devc-con-hid
        0         0         0         0      4096         0       4103 devc-con-hid (thread 2)
        0         0         0         0     20480         0       4103 devc-con-hid (thread 1)
        0    102400      8192         0         0         0       4103 devc-con-hid (proc/boot/devc-con-hid)
        0         0     16384         0         0         0       4103 devc-con-hid (proc/boot/libc.so.3)
        0         0         0         0         0  (   36864)     4103 devc-con-hid (/dev/mem)
```

From the breakdown details, we can see:

- This process has two threads. The stack size of thread 2 is 4096 bytes (4 KB), and the stack size of thread 1 is 20480 bytes (20 KB).

Copyright © 2014, QNX Software Systems Limited

- The private data of the shared library `libc.so.3` contributes 16384 bytes to the total memory usage of the process.
- This process refers to the memory `/dev/mem`, but this memory has already been accounted for in the total memory usage. Therefore, we don't include this size in the total memory usage of this process.

Copyright © 2014, QNX Software Systems Limited

showmount

Show remote NFS mounts on host

Syntax:

```
showmount [-ade] [host]
```

Runs on:

QNX Neutrino

Options:

-a

List all mountpoints in this form: host:*dirpath*

-d

Instead of printing the names of hosts, list the directory paths of mountpoints.

-e

Show the host's exports list.

host

The name of a host (default is this host).

Description:

The `showmount` utility shows status information about the NFS server on the specified host. By default, it prints the names of all hosts that have NFS filesystems mounted on that host.

Based on:

RFC 1094 (Appendix A: “Mount Protocol Definition”)

Caveats:

Because the NFS server is stateless, mount information is approximate. The `showmount` utility displays this information as accurately as the *nfsd* (p. 1348) daemon reports it.

Copyright © 2014, QNX Software Systems Limited

show_vesa

Display mode information for VESA BIOSes

You must log in as `root` and be in text mode to run this utility.

Syntax:

```
show_vesa
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

None.

Description:

The `show_vesa` utility displays information for VESA BIOSes up to VESA 3. This information includes general information, such as the amount of video RAM, as well as details about each supported video mode. This information is sent to *stdout*.

Examples:

Here's a sample of part of the output of `show_vesa`:

```
VESA Info Block

VESA Signature : VESA
VESA Version   : 0300
OEM String Ptr : 0080:0100
OEM String     : 3Dfx Interactive, Inc.
Capabilities   : 00000001
                 DAC width switchable to 8 bits per primary color
                 Controller VGA compatible
                 Normal RAMDAC operation
                 No Hardware Stereoscopic Signaling support
                 Stereo Signaling supported via external VESA stereo connector
Video Mode Ptr : 0080:0117
Total Memory   : 1000000 (16777216)
VBE Software Revision    : 0100
OEM Vendor Name Ptr      : 0080:015B
OEM Vendor Name String   : Elpin Systems, Inc.
OEM Product Name Ptr     : 0080:016F
OEM Product Name String  : 3Dfx Banshee
OEM Product Rev Ptr      : 0080:017C
OEM Revision String      : Version 1.00
OEM Data                 : 3Dfx Interactive, Inc.

DPMS Services available
Virtual Screen Services available

Total Number Of Video Modes found : 33

Mode : 0x0100 information
-------------------------
```

Copyright © 2014, QNX Software Systems Limited

```
Mode Attributes       : 009F
                        Mode Supported in Hardware
                        TTY Output functions supported by BIOS
                        Color Mode
                        Graphics Mode
                        VGA compatible
                        VGA windowed compatible
                        Linear Frame Buffer Available
                        Double Scan Mode Not Available
                        Interlaced Mode Not Available
                        No Hardware Triple Buffer Support
                        No Hardware Stereoscopic Support
                        No Dual Display Start Address Support
Window A Attributes   : 07
                        Relocatable Window(s)
                        Window is Readable
                        Window is Writeable
Window B Attributes   : 00
                        Single-Non Relocatable Window
                        Window is Not Readable
                        Window is Not Writeable
Window Granularity    : 0040
Window Size           : 0040
Window A Start Segment : A000
Window B Start Segment : 0000
Window Function Pointer: C000:2DAA
Bytes Per Scanline    : 0280 (640)
X Resolution          : 0280 (640)
Y Resolution          : 0190 (400)
X Char Size           : 08 (8)
Y Char Size           : 10 (16)
Number Of Planes      : 01 (1)
Bits Per Pixel        : 08 (8)
Number Of Banks       : 01 (1)
Memory Model          : 04 - Packed Pixel
Bank Size             : 00
Number Of Image Pages : 40 (64)
Reserved              : 01
Flat Frame Buffer Addr : E2000000
Bytes Per Scan Line (Linear)                : 0280 (640)
Number of Images for Banked Modes           : 40 (64)
Number of Images for Linear Modes           : 40 (64)
Red Mask Size (Linear)                      : 00 (0)
Red Field Position (Linear)                 : 00 (0)
Green Mask Size (Linear)                    : 00 (0)
Green Field Position (Linear)               : 00 (0)
Blue Mask Size (Linear)                     : 00 (0)
Blue Field Position (Linear)                : 00 (0)
Reserved Mask Size (Linear)                 : 00 (0)
Reserved Field Position (Linear)            : 00 (0)
Maximum Pixel Clock                         : 09896800 (160000000Hz)
```


Copyright © 2014, QNX Software Systems Limited

shutdown

Shut down and reboot the system (QNX Neutrino)

You must be `root` to run this utility.

Syntax:

```
shutdown [options]
```

Runs on:

QNX Neutrino

Options:

-b

Shut down but don't reboot. You can't use this option with -n *nodename*.

-f

Shut down fast. Reduce the amount of time between sending a `SIGTERM` signal and a sending a `SIGKILL` to processes that catch `SIGTERM`.

-n *nodename*

Shut down the specified node (default is current node).

-q

Be quiet.

-S *type*

The type of shutdown, which must be one of:

- `system` — shut down the system.
- `reboot` — reboot the system.

The default is `reboot`.

-v

Be verbose.

Copyright © 2014, QNX Software Systems Limited

Description:

The `shutdown` utility performs an orderly system shutdown as follows:

1. It sends a `SIGTERM` signal to all processes listed under `/proc`.
2. It then waits for a period of time (reduced if you specify the -f option).
3. It sends a `SIGKILL` signal to all remaining processes.
4. It reboots the system (unless you specified the -b option).

The interval between the `SIGTERM` and `SIGKILL` signals allows processes that have elected to catch the `SIGTERM` signal to perform any cleanup they need to do before the system is rebooted.

Files:

`/var/log/wtmp`

If this file already exists, `shutdown` adds an entry to it before shutting down or rebooting the system.

The `shutdown` utility doesn't create `/var/log/wtmp` if it doesn't already exist. This file can quickly become very big, which isn't good on an embedded system with limited resources.

Copyright © 2014, QNX Software Systems Limited

size

List section and total sizes for an archive or object file (POSIX)

Syntax:

size_variant [*options*] [*objfile*…]

where *size_variant* depends on the target platform, as follows:

Target platform	*size_variant*
ARMv7	`ntoarmv7-size`
x86	`ntox86-size`

Runs on:

Linux, Microsoft Windows

Description:

The `size` utility lists the section sizes and the total size for each of the object or archive files *objfile* in its argument list. By default, one line of output is generated for each object file or each module in an archive.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

slattach

Attach serial lines as network interfaces

Syntax:

```
slattach [-Hhlmn] [-s baudrate] [-t ldisc] ttyname
```

Runs on:

QNX Neutrino

Options:

-H

Turn on DTR/CTS flow control. By default, no flow control is done.

-h

Turn on RTS/CTS flow control. By default, no flow control is done.

-l

Turn on the CLOCAL flag, making it possible to run SLIP on a cable without modem control signals (e.g., DTR, DSR, DCD).

-m

Maintain modem control signals after closing the line. Specifically, this disables HUPCL.

-n

Don't detach from invoking tty.

-s *baudrate*

The speed of the connection. If not specified, the default of 9600 is used.

-t *ldisc*

Specifies the line discipline to use for the tty. The only currently support line discipline is `slip` (the default), which creates an SL instance.

ttyname

The name of the TTY device, a string in the form `tty`*XX* or `/dev/tty`*XX*.

Copyright © 2014, QNX Software Systems Limited

Description:

You can use `slattach` to assign a TTY line to a network interface that uses asynchronous serial lines.

Currently you use `slattach` to attach an SL instance (see *`lsm-slip.so`* (p. 1157)). You have to create an interface, using the *`ifconfig create`* (p. 957) subcommand, before using `slattach`. You can then configure the network source and destination addresses and other interface parameters via `ifconfig`.

Only the superuser may attach a network interface.

To detach the interface, use `ifconfig` *interface-name* down after killing off the `slattach` process. The *interface-name* is the name that's shown by *`netstat`* (p. 1339).

There's no way to specify the interface name (`sl%d`, etc.) to be attached by the `slattach` command. There's also no way to see which interface is assigned to the specified TTY by the `slattach` command.

Diagnostics:

Messages indicating that the specified interface is not configured or created, the requested address is unknown, or that the user is not privileged but tried to alter an interface's configuration.

Examples:

```
mount -Tio-pkt lsm-slip.so
ifconfig sl0 create
slattach -t slip -s 115200 /dev/serusb1
ifconfig sl0 inet 192.168.2.1 192.168.2.2

slattach ttyh8

slattach -s 4800 /dev/tty01
```

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

slay

Kill or modify a process by name or ID (QNX Neutrino)

Syntax:

```
slay [options]... process_name|process_id ...
```

Runs on:

QNX Neutrino

Options:

- *signal_number*

A signal number specifying which signal to raise on the processes matching *process_name* or *process_id*. For a list of signal numbers, see <signal.h>.

-C *cpunum*

(QNX Neutrino Core OS 6.3.2 or later) Set the CPU affinity to *cpunum*, where the first CPU is 0. You can use this option multiple times. For more information, see "*Setting the runmask* (p. 1777)", below.

-f

Force the action to be taken on all processes sharing the same *process_name* or ID. Normally, `slay` prompts for confirmation when more than one process bears the specified name or ID.

-h

Set a `SIGSTOP` signal on a process, effectively holding its execution.

-i

(QNX Neutrino Core OS 6.3.2 or later) Set the inherit mask in addition to the runmask, when used in conjunction with the -C or -R option. For more information, see "*Setting the runmask* (p. 1777)", below.

-m namelpid

Restrict the match to only the process name or to only the process ID. By default, `slay` matches both process IDs and names. For example:

This command:	Matches a process with:
`slay 1234`	ID 1234 or the name 1234

Copyright © 2014, QNX Software Systems Limited

This command:	Matches a process with:
`slay -m pid 1234`	ID 1234
`slay -m name 1234`	The name `1234`

-n *nodename*

(QNX Neutrino Core OS 6.3.2 or later) Search for the specified processes on the specified remote node *nodename*.

-P *prio*[flrlo]

Set the priority of the processes matching *process_name* to *prio*. Non-`root` users are limited to a maximum priority of 63; `root` can specify a priority up to 255. You can change the range of privileged priorities with the -P option for *`procnto`* (p. 1586).

The priority may be followed by f, r or o to change the scheduling policy to SCHED_FIFO, SCHED_RR or SCHED_OTHER, respectively.

SCHED_OTHER is currently the same as SCHED_RR.

If you specify -P without -T, `slay` sets the priority of all threads in the specified process or processes.

-p

Print the process IDs, in decimal, to standard output, with one process ID per line. The processes aren't slain.

-q

Query before dealing with the process, even if only one process is found with a matching name or ID (overrides option -f). This option is useful for viewing the other process information that `slay` presents.

-Q

Be quiet. This option is useful when you invoke `slay` from a C program.

-R *runmask*

(QNX Neutrino Core OS 6.3.2 or later) Set the CPU affinity to *runmask*. You can use this option multiple times to specify masks that are more than 32 bits long. For more information, see "*Setting the runmask* (p. 1777)," below.

-S

Copyright © 2014, QNX Software Systems Limited

Don't kill processes that have child processes. You typically use this option in a shell command that shuts down shells on other devices. Setting this option prevents `slay` from killing shells that have other processes (such as editors) running. If you also specify -q, `slay` prompts for a forced kill even if the named process has child processes.

-s *signal_name*

The signal to send. This option causes the signal *sig* to be raised for the processes matching the *process_name* or ID.

-T *tid*

(QNX Neutrino Core OS 6.3.2 or later) Apply the action to the thread with the given ID. You can use this option to direct a signal to a specific thread, or to change a thread's priority or runmask.

-t *ttyname*

Match only those processes whose name (or ID) is *process_name* (*process_id*) and have *ttyname* as the controlling terminal. If *ttyname* doesn't begin with a slash (`/`), `slay` assumes that it starts with the `/dev/` prefix.

-u

Set a `SIGCONT` signal on a process. If execution of the process was being held by a `SIGSTOP` signal, execution begins where it left off. If the process hadn't previously had a `SIGSTOP` set upon it, the `SIGCONT` signal has no effect.

-v

Be verbose; display messages about processes being signaled.

process_name

The name of a process to operate on.

process_id

The ID of a process to operate on.

Description:

Use the `slay` utility to kill or modify a process by name or by ID. Process names are specified without the path. For example, let's say you have a process called `/bin/sleep` that you want to kill. Entering `sleep` as the process name is sufficient to allow `slay` to find and kill it.

Copyright © 2014, QNX Software Systems Limited

There are many forms of this command. The simplest and most often used form is:

```
slay process_name|process_id
```

This command locates the process bearing the specified name or ID. If only one is found, a `SIGTERM` signal is set on it. If more than one process bears the specified name or ID, you're prompted for a yes/no response for each process. When each process is listed in this form, the process name, *pid*, and tty group/member numbers are also displayed to help you make a selection.

To set a signal on a process you must either own the process or be logged in as `root`.

Setting the runmask

On a multicore system, you can use a runmask to specify which processors a thread or process can run on. The default is all 1s (i.e. all CPUs).

The runmask is useful only on multiprocessor systems.

You can use `slay` to change the runmask, or the runmask and inherit mask, for threads that are already running; to set the masks for a new process, use the *on* (p. 1417) command. Both commands interpret the -C and -R options in the same way.

You can use more than one -R option to specify a runmask that's more than 32 bits long. The first -R option specifies bits 0 through 31, the second specifies bits 32 through 63, and so on.

If you use both the -C and -R options or multiple instances of them, the resultant mask is the bitwise ORing of all -C and -R options. For example, `slay -R 0x1` is equivalent to `slay -C0`, and `slay -R 0x1 -C3` is equivalent to `slay -C0 -C3`.

When you use -R or -C without -T, all threads in the specified process or processes are affected.

If you use -R or -C, `slay` always changes the runmask for the specified threads or processes. If you also specify the -i option, `slay` also sets the inherit mask to the same value as the runmask.

- If the resulting runmask specifies at least one CPU that's physically present, the runmask is accepted, and any bits that correspond to CPUs that aren't present are ignored. If the resulting runmask doesn't specify any CPU that are physically present, an error results.
- If you change the runmask for a process, the processor for blocked threads doesn't change until the threads become unblocked (or never if the threads remain blocked).

Copyright © 2014, QNX Software Systems Limited

For more information about runmasks, see the Multicore Processing chapter of the *System Architecture* guide, and the Developing Multicore Systems chapter of the Multicore Processing *User's Guide*.

Examples:

Kill the `spooler` process on node `peterv`:

```
slay -n peterv spooler
```

As `root`, change priority of the `my_test` process to 20:

```
slay -P 20 my_test
```

Exit status:

0

No processes matched the supplied criteria, an error occurred, or the number of processes matched and acted upon was an even multiple of 256.

1-128

The number of processes matched and acted upon modulo 256 (e.g. a status of 1 could mean 1 process, 257 processes, 513 processes etc.)

129-160

If the exit status was gleaned through direct spawning, this is the number of processes matched and acted upon modulo 256. If `slay` was run through the shell, this is either the number of processes matched and acted upon, or it indicates why `slay` died due to a signal (subtract 128 from the exit status to determine the signal number).

161-255

The number of processes matched and acted upon, modulo 256.

Caveats:

The exit status of `slay` is nonstandard for historical reasons. We strongly recommend that you not use `slay` in any situation where the exit status is relied upon because the status is ambiguous in some circumstances.

Copyright © 2014, QNX Software Systems Limited

sleep

Suspend execution for an interval (POSIX)

Syntax:

```
sleep time
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

time

For POSIX conformance, this must be a nonnegative decimal integer, from 0 to 4294967295, specifying the number of seconds to suspend execution.

As a QNX Neutrino extension, *time* can be a floating point number, so you can specify fractions of seconds.

Description:

The `sleep` utility suspends execution for at least the number of seconds specified by the *time* operand.

Exit status:

0

The execution was successfully suspended for at least *time* seconds, or a `SIGALRM` signal was received.

>0

An error occurred.

Caveats:

The `sleep` utility takes the default action for all signals, except that `sleep` terminates normally with a zero exit status on receipt of a `SIGALRM` signal.

Copyright © 2014, QNX Software Systems Limited

slinger

Tiny webserver

Syntax:

```
slinger [-a administrator] [-c] [-d] [-e]
        [-i address] [-n] [-p port_number] [-s]
```

Runs on:

QNX Neutrino

Options:

-a *administrator*

Contact information of the administrator of Slinger (e.g. `-a my_admin@something.com`). The information is stored in the ***SERVER_ADMIN*** (p. 1794) environment variable. The default is `nobody`.

-c

Request that HTML files with SSI not be placed in the browser cache. Instead of using a previously downloaded version from its local cache, the browser contacts the server whenever it displays an HTML file containing SSI. Use this option when you'd like the browser to display the latest version of your dynamic SSI data.

-d

Write debugging info to the system log. In order to capture the log messages, you need to have *syslogd* (p. 1885) running.

-e

Enable the SSI `exec` command.

-i *address*

The interface for `slinger` to listen on (e.g. 10.0.0.1). The default is any.

-n

Don't move to the background.

-p *port_number*

Set the port number.

Copyright © 2014, QNX Software Systems Limited

-s

Enable the *SSI commands* (p. 1782).

Description:

The `slinger` utility is an HTTP server for resource-constrained environments. Slinger is compliant with the CGI 1.1 and HTTP 1.1 standards, and provides support for dynamic HTML via Server Side Includes (SSI).

Running Slinger

To run Slinger, type:

```
slinger &
```

The `slinger` server listens on the TCP port 80. Since this is under 1024, `slinger` needs to be run as `root`. As soon as it has attached to the HTTP port, it drops itself down to user -2 (`setuid (-2)`).

How dynamic HTML works

Slinger supports Server Side Include (SSI) commands to provide dynamic HTML. For example, Slinger uses the ***PATH*** and ***CMD_INT*** environment variables to provide information to the SSI command, *exec* (p. 1783). Using dynamic HTML, clients can offer interactive realtime features on their web pages.

Clients code dynamic HTML by placing *SSI tokens* (p. 1781) in the HTML code of their web pages. The SSI token contains an SSI command that's handled by Slinger. While transmitting the HTML code, Slinger replaces a token with HTML data based on the *tag* (p. 1782) contained in the SSI token.

For Slinger to process SSI tokens, the HTML file must have `.shtml` as its file extension.

Syntax of an SSI token

The syntax of an SSI token is as follows:

```
<!--#tag [[variable_set[="value"]] …] -->
```

where:

<!--

Starting delimiter of an SGML/HTML comment.

#

Tells Slinger the comment contains a command.

tag

Copyright © 2014, QNX Software Systems Limited

SSI command.

variable_set

One or more arguments to the SSI command.

value

One or more values for the variable set.

-->

Ending delimiter of SGML/HTML comment.

There must be a space before the ending comment delimiter.

There's no limit to the token size (other than memory).

SSI commands

The valid SSI commands are: `break`, `config`, `echo`, `exec`, `if`, `goto`, `include`, `label`, and `qnxvar`.

`break`

Terminate transmission of an HTML document at any point. The `break` command has no arguments, so the syntax is:

```
<!--#break -->
```

`config`

Set certain HTML output options. The syntax is:

```
<!--#config variable_set="value" -->
```

The *variable_set* may contain: cmdecho, cmdprefix, cmdpostfix, errmsg, onerr, timefmt. For details, see the "*config commands* (p. 1788)" section.

`echo`

Insert data from an HTML form field or an environment variable onto an HTML page. The `echo` command has one argument, var. The syntax is:

```
<!--#echo var="string" -->
```

where *string* is the value of an HTML form field or an environment variable. For example:

```
<!--#echo var="Last name" -->
```

Or

```
<!--#echo var="DATE_LOCAL" -->
```


Copyright © 2014, QNX Software Systems Limited

The echo command uses these SSI variables, as well as *CGI environment variables* (p. 1794) (for more information, see the Environment variables section in this utility):

DATE_GMT

Current Greenwich Mean date and time.

DATE_LOCAL

Current local date and time.

DOCUMENT_NAME

Complete local pathname of the current web page.

DOCUMENT_URI

Local pathname of the current web page referenced to the base directory of the web space.

LAST_MODIFIED

Date and time of the last modification of the current web page.

QUERY_STRING_UNESCAPED

Unescaped query string sent by the remote client. All characters that are special to the shell are escaped with \.

exec

Spawn an external program and print the output at the location of the exec token on the HTML page. The external program is spawned to increase performance. The exec command takes the following arguments:

- cgi — Path and filename of a CGI script. The syntax is:

```
<!--#exec cgi="executable[/pathinfo]" -->
```

where:

executable

Name of a CGI executable.

pathinfo

Path information provided to the *executable*. (The ***PATH*** environment variable can also contain path information provided to CGI scripts.)

For example, suppose you have a CGI script, othello.cgi, and the ***HTTPD_SCRIPTALIAS*** (p. 1793) environment variable contains the directory

Copyright © 2014, QNX Software Systems Limited

`/web/games/`. If you put the following token on your HTML page, Slinger searches for the `othello.cgi` script in the `/web/games/` directory:

```
<!--#exec cgi="othello.cgi" -->
```

- cmd — Spawn a shell and run a shell executable command. The syntax is:

  ```
  <!--#exec cmd="[path/]executable [args]" -->
  ```

  where:

  [path/]executable

  Name of an executable command.

  args

  List of command-line arguments.

  The `cmd` token uses the ***CMD_INT*** (p. 1793) environment variable to identify the command interpreter, *shell_program*, and then spawns a shell using the following:

  ```
  shell_program -c "string"
  ```

  where *string* is [*pathname/*]*executable* [*arguments*].

  If a full pathname isn't provided, Slinger uses the path defined in the ***PATH*** (p. 1793) environment variable to locate the *shell_program* and the *executable*. You can echo the output using:

  ```
  <!--#config cmdecho (p. 1788)="ON" -->
  ```

  For example:

  ```
  <!--#config cmdecho (p. 1788)="ON" -->
  <!--#exec cmd="cd /home/smr; ls" -->
  ```

if

Conditionally execute SSI operations and conditionally print the HTML text. The syntax of the `if` command is:

```
<!--#if "op1" operator "op2" operation -->
```

where:

op1

A string that's the first operand of a logical comparison statement.

op2

A string that's the second operand of a logical comparison statement.

Copyright © 2014, QNX Software Systems Limited

operator

One of `==` `!=` `<` `>` `!<` `!>` `hasstring`

operation

Action to take if the logical comparison evaluates to TRUE, where the possible actions are: `break`, `error`, `errorbreak`, `goto`, `print`, and `printbreak`.

The `hasstring` operator returns TRUE if the character string in *op2* is found in the *op1* string.

If both operands are numbers, they're compared as numbers; otherwise, the comparison is based on the alphabetic order of the operands.

If the logical comparison evaluates to FALSE, nothing happens. For example:

```
<!--#if "5" != "10" goto thislabel -->
```

The NULL operand is defined by `""`. NULL may be used to check for the existence of data in a form field.

For example, if you have a form with the field `Last name` and you want to ensure the user doesn't leave it blank, then the token is:

```
<!--#if "Last name" == "" printbreak "<P>The Last
 name field cannot be blank; please resubmit the
 form." -->
```

If the field isn't empty, nothing happens. But if the field is empty, the message is displayed and the HTML document terminates.

goto

Jump to a labeled token (without printing skipped text or applying skipped tokens). The syntax of the `goto` command is:

```
<!--#goto ="label_name" -->
```

where *label_name* is defined in a subsequent `label` command on the HTML page.

Remember to put a space before the equal sign.

Here's an example of how the `goto` and `label` commands are used together:

```
<!--#goto ="change_temp" -->
<!--Skip this part of the HTML page. -->
<!--#label ="change_temp" -->
<!--Slinger continues scanning -->
<!--the HTML code here. -->
```

include

Insert the contents of a file into an HTML page. The `include` command takes these arguments:

Copyright © 2014, QNX Software Systems Limited

- file — The syntax is:

  ```
  <!--#include file="pathname" -->
  ```

  where *pathname* is the path and filename of an HTML file relative to the directory of the current web page. Don't use absolute paths, and don't use `../` in the pathname (since you can't include a pathname that begins above the current directory).

  Use the file tag when you include files that are in the same directory as the current directory. For example, suppose you have a directory `web/support/docs/` containing two files `my_doc.html` and `docs.html`, and the URL is `www.something.com/support/docs/docs.html`. To insert the `my_doc.html` file into the HTML page, write the token as:

  ```
  <!--#include file="my_doc.html" -->
  ```

  The file `my_doc.html` is relative to the directory (`web/support/docs`) of the current document (`docs.html`).

- virtual — The syntax is:

  ```
  <!--#include virtual="pathname" -->
  ```

  where *pathname* is the path and filename relative to the root directory on the Slinger server, configured in the ***HTTPD_ROOT_DIR*** environment variable.

  When you use the virtual tag, Slinger begins its search for the file from the root directory. For example, suppose you have a directory `web/support/docs/` that contains the file `my_doc.html`, and the ***HTTPD_ROOT_DIR*** environment variable contains the `web` directory. To insert `my_doc.html` into the HTML page with the URL `web/support/docs/my_doc.html`, write the token as:

  ```
  <!--#include virtual="/support/docs/my_doc.html" -->
  ```

The `include` command is recursive; each file inserted may contain `include` tokens.

Although the inserted file can't be a CGI script, it can contain a reference to a CGI script.

`label`

When Slinger encounters a *goto* (p. 1785) command, jump to the `label` token on the HTML page. The syntax of the `label` command is:

```
<!--#label ="label_name" -->
```

The maximum length of *label_name* is 60 characters. Remember to put a space before the equal sign.

Copyright © 2014, QNX Software Systems Limited

For example:

```
<!--#label ="change_temp" -->
```

qnxvar

Get data or change data on the data server process. The `qnxvar` command takes the following arguments:

- format — Set the format for subsequent displays of data obtained from the *data server* (p. 1790) process, on the HTML page. The syntax is:

  ```
  <!--#qnxvar format="[text] %s [text] " -->
  ```

  where:

  %s

  A placeholder for the value of the variable.

  text

  Text that can contain HTML tags, for example <P>.

  The format remains in effect until the next format statement. For example:

  ```
  <!--#qnxvar format="I work for %s." -->
  ```

  If you want to print the % sign followed by a variable, you have to enter: \% %s. To print %s, type \%s.

- read — Get the value of a variable on the data server process and display the result at the location of the `qnxvar` token on the HTML page. The syntax is:

  ```
  <!--#qnxvar read="var_name  len [var_name  len]" -->
  ```

  where *len* is the size in bytes.

  Data is printed up to `NULL` or the length of the data. The format of the result is specified with the format tag. For example:

  ```
  <!--#qnxvar read="company_name 20" -->
  ```

- write — Change a variable on the data server process. The syntax is:

  ```
  <!--#qnxvar write="var_name "data"" -->
  ```

  where *data* is a string. For example:

  ```
  <!--#qnxvar write="set.temperature "300"" -->
  ```

Copyright © 2014, QNX Software Systems Limited

The maximum length of *var_name* is 60 characters. Don't omit the space after the *var_name*.

config commands

cmdecho [ON | OFF]

Set the output option of subsequent *exec* (p. 1783) tokens. The default is OFF, meaning the output isn't parsed or printed. The cmdecho command doesn't apply to the output of CGI scripts.

cmdprefix

Set a prefix to each line output from subsequent *exec* (p. 1783) tokens. The syntax is:

```
<!--# config cmdprefix="string" -->
```

where *string* is any character string or HTML format tag.

If cmdecho is set to ON, the output is prefixed with *string*. Otherwise, the output isn't parsed or printed.

cmdpostfix

Set the string appended to the end of each line output from *exec* (p. 1783) tokens. The syntax is:

```
<!--# config cmdpostfix="string" -->
```

where *string* is any character string or HTML format tag.

If cmdecho is set to ON, the output is appended with *string*. Otherwise, the output isn't parsed or printed.

For example, to make the output of the exec command appear on separate lines:

```
<!--# config cmdpostfix="<BR>" -->
```

errmsg

Compose an error message and print it when Slinger encounters an SSI error, such as a parsing error or unavailable required data. The syntax is:

```
<!--# config errmsg="string" -->
```

where *string* is any character string or HTML format tag. For example:

```
<!--# config errmsg="<P><CENTER>*ERROR*</CENTER>" -->
```


Copyright © 2014, QNX Software Systems Limited

The error message remains in effect until the next errmsg is configured.

onerr

Set the action to be taken when Slinger encounters an error. The syntax is:

```
<!--#config onerr="action" -->
```

The possible values of *action* are:

- `goto` —Jump to a *labeled* (p. 1786) token. The syntax is:

  ```
  <!--#config onerr="goto label" -->
  ```

  For example:

  ```
  <!--#config onerr="goto oven_temp" -->
  ```

- `break` — Terminate transmission of the HTML document to the client.
- `error` — Print the current error message specified by errmsg.
- `errorbreak` — Print the current error message specified by errmsg and terminate transmission of the HTML document to the client.
- `print` — Print text. The syntax is:

  ```
  <!--#config onerr="print " string"" -->
  ```

  where *string* is any character string or HTML tag.

- `printbreak` — Print text and terminate transmission of the HTML document to the client. The syntax is:

  ```
  <!--#config onerr="printbreak " string"" -->
  ```

  where *string* is any character string or HTML tag. For example:

  ```
  <!--#config onerr="printbreak "<P>printbreak
   error encountered. Terminating transmission of HTML
   text."" -->
  ```

timefmt

Set the format of the date and time. The syntax is:

```
<!--#config timefmt ="string" -->
```

where *string* is compatible with the ANSI C library function, *strftime()*. The default string is `%a %b %d %T %Y`.

For example:

```
<!--#config timefmt ="%A %B %d %Y" -->
```

Copyright © 2014, QNX Software Systems Limited

Ways to achieve dynamic HTML

You can dynamically generate HTML in order to support changing systems. The main components in such a system are:

- the changing element
- external applications that activate and read change
- the data server process
- Slinger
- the remote clients.

There are three ways to provide dynamic HTML:

- CGI interface. This is described in many commonly available books.
- Write an I/O manager to provide changing HTML each time the prefix is opened/read.
- Use Server Side Includes with the data server. The rest of this section describes this method.

The data server

The data server process, *ds* (p. 645), stores global data. External applications can modify or read this global state through the use of the data server API. Slinger can modify or read the global state via the `qnxvar` SSI command.

Here's a conceptual view of a device monitored and controlled by both an external application and a remote client of Slinger:

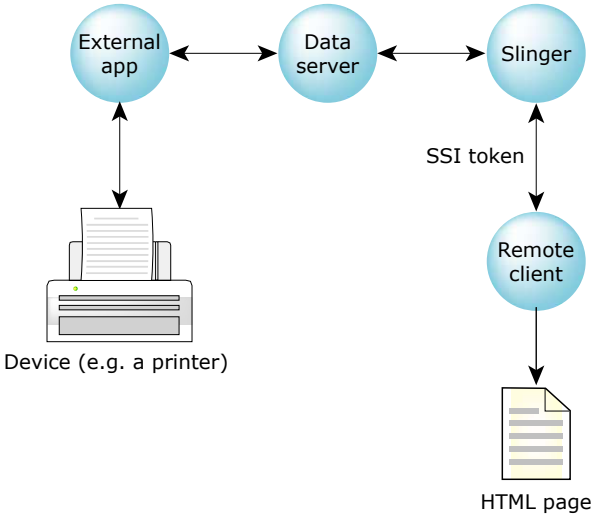


Figure 8: Using the data server.

To use dynamic HTML:

1. Use the SSI tokens to manipulate the HTML page.
2. Use the *qnxvar* (p. 1787) tokens to query and manipulate the global data.
3. Write an application to manipulate the global state using the data server library functions. See the examples in the *ds* (p. 645) utility.

Copyright © 2014, QNX Software Systems Limited

Executing CGI scripts

If you want your CGI script to be executed, then the script must be requested with the URL:

`http://`*`hostname`*`/cgi-bin/`*`scriptfile`*

where:

hostname

Name of the host or the IP address of the host running Slinger.

`cgi-bin`

Execute the CGI script file specified.

scriptfile

Name of the CGI script to be executed. Slinger uses the environment variable ***`HTTPD_SCRIPTALIAS`*** to locate the pathname of the CGI script file.

When Slinger executes a CGI script, it parses the output of the script for HTTP header directives. Slinger must buffer the header directives, before transmitting the script data, in case a header directive is passed that'll affect the format of the default header directives. Up to 1K of header information can be buffered. CGI scripts must provide a valid header.

Slinger identifies the end of the header as a blank line, lines are terminated with a <LF> or a <CR><LF>. A common HTTP header directive to provide specifies the type of data that the CGI script provides. The default Content-Type is “text/html.” For example:

```
Content-Type: text/html<LF>
<LF>
CGI script data...
```

Slinger supports the following CGI header directives:

Directive	Description	Example
Content-Type	Type of data being sent back to the client.	Content-Type: text/html
Location	Reference a file by a URL.	Location: `http://www.qnx.com`
Status	Pass a status code and explanation back to the client. If Slinger executes the script successfully, it passes a status of `200 OK`	Status: 200 OK

Copyright © 2014, QNX Software Systems Limited

Directive	Description	Example
	(unless the script changes the status).	

Any other directives are just added to the HTTP header.

A script with the filename prefix `nph-` (e.g. `nph-myscript.cgi`) won't be parsed by Slinger; only the raw data from the script is sent. Because Slinger doesn't add any HTTP header content, any script that isn't parsed must provide the entire HTTP header in its script output.

Security precautions

When you choose the directory for your data files:

- Don't place any sensitive files in the document directory.
- Isolate your data files directory from the system files directory. For example, `/usr/www` is much safer than the root directory, `/`. The root directory `/` opens up your whole system to be served by Slinger.

If you configure Slinger to support CGI:

- Place the CGI scripts in a directory isolated from your normal system binaries. Don't use `/bin` or `/usr/bin` as your CGI directory.
- Avoid setting your CGI script file permissions to “set user ID when executing” when the file is owned by a privileged user (e.g. `root`).
- Keep your CGI scripts and documents in separate directories. This prevents people from accessing your scripts.

Don't expose your machine to undue risk. Make sure that:

- The permissions on all the files and directories are read-only.
- No files are owned by userid `(-2)` because Slinger runs as userid `(-2)` and you don't want Slinger to own the files.

These precautions prevent somebody from giving your machine a new password file or tampering with your web pages.

Based on:

RFC 2068

Examples:

The right way

Copyright © 2014, QNX Software Systems Limited

We recommend that you put your documents and scripts in separate directories. In this example, the documents are in the `/usr/webdoc` directory, the root document is `foo.html`, and the CGI scripts are in `/usr/web/cgi`:

```
export HTTPD_ROOT_DIR=/usr/webdoc
export HTTPD_ROOT_DOC=foo.html
export HTTPD_SCRIPTALIAS=/usr/web/cgi
slinger &
```

The "wrong" way

In this example, anyone can download the scripts because the documents and scripts are in the same directory:

```
export HTTPD_ROOT_DIR=/usr/web
export HTTPD_ROOT_DOC=foo.html
export HTTPD_SCRIPTALIAS=/usr/web
slinger &
```

Files:

The `slinger` webserver requires a TCP/IP stack (included in *io-pkt** (p. 1007)) and the `libsocket.so` shared libraries.

Environment variables:

Slinger configuration

Slinger uses the following environment variables to set its configuration:

PATH

The pathname of the *command interpreter* (p. 1783) and shell executable command used in the SSI command, `exec`. For more information, see the *exec cmd* (p. 1783) description.

CMD_INT

The pathname or name of the shell program used to interpret the string passed to the SSI command, *exec cmd* (p. 1783) (e.g. `/bin/sh`). If the pathname isn't given, Slinger searches the pathname in ***PATH*** for the shell program.

HTTPD_ROOT_DIR

The name of the directory where Slinger looks for data files. The default is `/usr/local/httpd`.

HTTPD_ROOT_DOC

The name of the root document (e.g. `index.html`). When a web client requests the root document, ***HTTPD_ROOT_DOC*** is appended to

Copyright © 2014, QNX Software Systems Limited

HTTPD_ROOT_DIR to build the full pathname of the root document. The default is `index.html`.

For example, let's say ***HTTPD_ROOT_DOC*** is defined as `index.html` and ***HTTPD_ROOT_DIR*** is defined as `/usr/www`. Slinger appends `index.html` to `/usr/www` to build `/usr/www/index.html`.

HTTPD_SCRIPTALIAS

The directory where Slinger looks for the CGI executables.

Note that if you define ***HTTPD_SCRIPTALIAS***, anybody can run scripts or processes that reside in this directory on your machine. Not defining ***HTTPD_SCRIPTALIAS*** turns CGI off, causing all CGI requests to fail.

For example, suppose ***HTTPD_SCRIPTALIAS*** contains `/cgi/bin` as the name of the directory. If Slinger gets a request for the resource `www.qnx.com/cgi-bin/get_data.cgi/foo`, the `get_data.cgi` script found in `/cgi/bin` is executed, and `foo` is sent as pathname information to `get_data.cgi`. The `foo` directory is stored in the ***PATH_INFO*** environment variable (described below).

To help keep your system secure, don't use `/bin` or `/usr/bin` as your CGI directory.

Available to CGI scripts

Slinger sets the following environment variables, which can be used by CGI scripts:

ACCEPT_LANGUAGE

The languages that can be read by the remote client.

CONTENT_LENGTH

Length of attached information in the case of a POST.

CONTENT_TYPE

Type of attached information in the case of a POST.

DOCUMENT_ROOT

Location of data files. (Same as ***HTTP_ROOT_DIR***.)

FORWARDED

Name of the proxy server through which the web page is being processed.

Copyright © 2014, QNX Software Systems Limited

FROM

Name of the remote client user.

GATEWAY_INTERFACE

Name and version of the Common Gateway Interface served on Slinger.

HTTP_ACCEPT

MIME types that the client accepts, as given by HTTP headers.

HTTP_USER_AGENT

The browser that the client is using to send requests.

PATH

Pathname of the *command interpreter* (p. 1783) and shell executable command used in the SSI command, `exec`.

PATH_INFO

Extra path information sent.

PATH_TRANSLATED

Append the pathname in ***PATH_INFO*** to the pathname in ***HTTPD_ROOT_DIR***.

QUERY_STRING

Raw query string sent from the remote client.

REFERER

URL of the HTML page that referred the remote client to this web page.

REMOTE_ADDR

IP address of the remote client.

REMOTE_HOST

Hostname of the remote client.

REMOTE_IDENT

Remote username if supporting *RFC 931* identification.

REMOTE_PORT

Port of the remote client.

REMOTE_USER

Username used to validate authentication from the remote client.

Copyright © 2014, QNX Software Systems Limited

REQUEST_METHOD

Method by which the current web page was requested (GET or POST).

SCRIPT_NAME

Name of the script being executed.

SERVER_ADMIN

Contact information of the administrator of Slinger specified by Slinger's -a option.

SERVER_NAME

Name of the computer where Slinger is running.

SERVER_PORT

IP port that Slinger is answering on.

SERVER_PROTOCOL

Name and version of HTTP served on Slinger.

SERVER_SOFTWARE

Name of Slinger software.

SERVER_ROOT

Current working directory of Slinger.

TZ

Time zone.

Caveats:

The `slinger` webserver communicates over TCP sockets, so you need to have socket runtime support. This means you need to have a TCP/IP stack running.

When configuring `slinger`, take care not to expose your machine to undue risk:

- The directory containing the documents shouldn't have any sensitive files in it.
- The permissions should be read-only on all the files and directories.
- The ownership of the files *shouldn't* be user -2(32767).

These precautions prevent somebody from giving your machine a new password file, or tampering with your web pages.

If you configure `slinger` to support CGI, keep the following in mind:

- *Anyone* can run processes on your machine.

Copyright © 2014, QNX Software Systems Limited

- The CGI scripts should all be in a directory isolated from your normal system binaries. In other words, don't use the `/bin` or `/usr/bin` directories for your CGI directory!

Besides these precautions, you should also keep your documents and scripts in separate areas. This prevents people from either reading or modifying your scripts.

Copyright © 2014, QNX Software Systems Limited

slm

System launch and monitor: launch complex applications consisting of many processes that must be started in a specific order

Syntax:

```
slm [-aCkvV] [-D debug_mode] [-n subsystem_path] [-p priority]
    [-P search_path] [-r recovery_mode] [-R recovery_times]
    [-s comp_name] [-t polling_interval] [-T total_wait]
    [-x comp_name] config_file
```

Runs on:

QNX Neutrino

Options:

-a

Adopt running daemon processes. Use this option to integrate SLM with an existing system where some server processes may already be running. If you place component entries for all relevant system processes in the configuration file, SLM will adopt these processes at startup as if it had launched them itself (and can thus control the processes via the command interface or restart them automatically if they terminate abnormally).

-C

In case of error, let `slm` continue to work rather than bail out.

-D *debug_mode*

Specify when to use the <SLM:debug> argument list (instead of the normal <SLM:args> list). One of: `cmd` (default), `startup`, or `always`. With `cmd`, the debug list is used only when the module is started with the *-d* (p. 1801) option. With `startup`, all components launched at startup (see the *-s* (p. 1801) option) will initially use the debug list, but will then honor the -d option of subsequent restarts. With `always`, the debug list is always used.

-k

Control whether PATHSPACE and DAEMON_DIED system events should be used to kick the normally poll-driven `waitfor` and `stop` handling. Since these events are also used by other processing, each event trigger may require an additional amount of unrelated activity, but generally this is faster than

Copyright © 2014, QNX Software Systems Limited

the default polling periods in detecting process state changes. Using these system events is enabled by default; specify -k if you don't want to use them.

-n ***subsystem_path***

Set the access point (default is `/dev/slm`) for client applications to write *control and query commands* (p. 1800).

-p ***priority***

Set the priority of the SLM server threads (default is 30).

-P ***search_path***

Set the search path for executables (default is `$PATH`). When launching a process, SLM looks in the search path to find the executable if the corresponding `command` tag doesn't contain a full path.

-r ***recovery_mode***

Set the recovery mode for components monitored by SLM. One of: `none` (default), `stop`, or `replace`. The action specified with the -r option is performed when a component terminates abnormally if that component doesn't override this setting in its `repair` tag.

-R

Set the maximum number of times to attempt recovery (default is 2 times per minute).

-s ***comp_name***

Name a component or module to launch at startup. For convenience, you can use the built-in modules `"all"` and `"none"` (default is `"all"`).

-t ***polling_interval***

Set the polling interval in milliseconds for the `wait` property. Default is `100`.

-T ***total_wait***

Set the total wait time in milliseconds. Default is `50000`.

-v

Increase output verbosity (messages are written to `sloginfo`). The -v option is cumulative; each additional v adds a level of verbosity, up to 7 levels.

-V

Copyright © 2014, QNX Software Systems Limited

Log output messages to console (as well as `sloginfo`).

-x *comp_name*

Name a component or module to terminate at shutdown. For convenience, you can use the built-in modules `"all"` and `"none"` (default is `"all"`).

Description:

SLM is started from `startup.sh`, which runs during the boot sequence. In the SLM command line inside `startup.sh`, you must specify a configuration file, but all the other parameters are optional.

Control and query commands

Client applications can control SLM by writing commands to the `/dev/slm` interface.

Control commands can `start`, `stop`, `restart`, or `replace` a specified module or component. When you start a component, SLM will start any dependencies (that aren't already running) and wait for them as required. When you stop a component, SLM first stops any dependents on the component. Restarting is a sequential composition of stop and start operations and is typically applied to set a specific high-level module state. Replacing will stop and relaunch a component and then restart any currently active components that had a dependency on that component. This is typically applied to update a low-level component process.

Query commands can list the dependencies (`depend`), running components (`active`), or components that terminated abnormally (`dead`). Command lines consist of the command, any options, and a module or component name, if appropriate.

Only the system superuser (UID 0) can execute the control and query commands (except `active` and `depend`).

The following table summarizes the control and query commands:

Command	Options	Description
`active`	-v	List the active (running) components.
`dead`	-v -w	List the dead (faulted) components.
`depend`	-s -u -v	List dependencies or dependents for the specified component.

Copyright © 2014, QNX Software Systems Limited

Command	Options	Description
`start`	-d -v -x	Start the specified component.
`stop`	-s -v -x	Stop the specified component.
`restart`	-d -s -v -x	Stop then start the specified component.
`replace`	-d -s -v -x	Update the specified component.

The following table describes all the options:

Option	Description
-d	Debug mode: start components with their debug argument list.
-s	Stateless: ignore any stateless dependencies when stopping components.
-u	Used by: list components that depend on the specified component.
-v	Verbose: give details of each action performed when responding to a command.
-w	Wait: block until a process terminates abnormally.
-x	Explain: list the required actions but don't perform them.

Command example

Following execution of a command written to `/dev/slm`, the results are available to be read from the same file descriptor. Here's a simple example (with no error handling):

```
int    slm;
char   text[128];

slm = open("/dev/slm", O_RDWR);
write(slm, "start -v all", 12);
while (read(slm, text, sizeof(text)) > 0)
printf("%s\n", text);
close(slm);
```

Copyright © 2014, QNX Software Systems Limited

Issuing commands via the `slmctl` utility

Besides writing control/query commands programmatically, you can use the `slmctl` utility to send SLM commands (via the command line or typed interactively).

The utility displays the results of each action in a line describing the operation on the specified component or module as follows:

Utility output	Meaning
`START` *`component pid|error`*	Component was started.
`start` *`component`*	Component already active (no errors).
`WAIT` *`component error`*	Waiting for component.
`wait` *`component`*	Component already active (no errors).
`STOP` *`component error`*	Component stopped.
`stop` *`component`*	Component already inactive.
`BEGIN` *`module`*	Encapsulation of multiple components.
`END` *`module error`*	Reported only via `sloginfo`, not `slmctl`.

SLM configuration file

SLM uses an XML configuration file to determine the appropriate order for scheduling processes. The configuration file lists all the processes for SLM to manage, any dependencies between the processes, the commands for launching the processes, and other properties.

Configuration file structure

The root XML element of the configuration file is the `system` tag. All element names start with `SLM:`, so the root element (and the outline of the file) looks like this:

```
<SLM:system>
    -- component and module descriptions --
</SLM:system>
```

Components

A process managed by SLM is represented by a *component*. You must provide a component name (usually based on the process name) that will be used within the configuration file when specifying interprocess dependencies or membership in a group of components.

Copyright © 2014, QNX Software Systems Limited

All `component` tags are listed in the root element and contain other tags that describe the properties of individual components. The `component` tag syntax is as follows:

```
<SLM:component name="mcd">
    -- component properties --
</SLM:component>
```

The following table describes the `component` tags:

Tag	Attribute	Value(s)	Description
`args`		*command_args*	The list of command-line arguments to provide the binary executable.
`cd`		*dir_name*	The directory to switch into when launching the process; this directory becomes the process's working directory (`$CWD`).
`command`	`launch`	`[ bg[,nohup] ]`	Controls process creation.
		pathname	The full path of the binary executable (e.g., `/usr/bin/mcd`).
		`builtin`	The name of a built-in SLM command. Options are: • `mkdir`—creates one or more directories. List the directories to create in the `args` element. • `no_op`—does nothing, but allows waiting for a filepath. This mechanism can be used to detect whether a process started outside of SLM is ready. • `pathmgr_symlink`—creates one or more fast kernel symlinks. List the symlinks to create in the `args` element.
`debug`		*command_args*	An alternative list of command-line arguments to provide the binary executable when SLM is run in debug mode. This list might contain options (such as -v to increase verbosity).
`depend`	`state`	`[ session | stateless ]`	A component may need other services to be active before the component can run. Any prerequisites must be expressed as dependencies. There are two forms of dependency: *session* (stateful) and *stateless*. With session dependency (the default), a client/server relationship is assumed; the server stores state information on all its clients. In this model, if the server must be stopped or restarted, then all its clients must be stopped. With stateless dependency, the server doesn't maintain any client information, so it's not necessary to restart clients if the server is restarted.

Copyright © 2014, QNX Software Systems Limited

Tag	Attribute	Value(s)	Description
		component_name	Name of the prerequisite component. A component can have zero, one, or many dependencies.  You must define a separate tag for each dependency. SLM won't start a component until all the prerequisites are running.
`priority`		*priority_policy*	An alphanumeric value indicating the priority level and scheduling policy to assign the process (e.g., `10r`).
`repair`		`[ default | none | stop | replace ]`	Specifies the action to take if the component terminates abnormally: • `default`—tells SLM to perform the action specified by the -r command-line option • `none`—SLM takes no recovery action • `stop`—SLM stops any other components that depend on the component that failed • `replace`—SLM restarts the failed component
`stderr`	`iomode`	`[ w[+] | a[+] ]`	The access mode: overwrite (`w`), read and overwrite (`w+`), append (`a`), or read and append (`a+`).
		filename	Name of the file for redirecting standard error (`stderr`).
`stdin`	`iomode`	`[ r[+] ]`	The access mode: read only (`r`) or read and write (`r+`).
		filename	Name of the file for redirecting standard input (`stdin`).
`stdout`	`iomode`	`[ w | a ]`	The access mode: overwrite (`w`) or append (`a`).
		filename	Name of the file for redirecting standard output (`stdout`).
`stop`	`stop`	`[ none | signal ]`	The `signal` setting (the default) causes SLM to send a signal to the underlying process. The `none` setting disables the signaling; in this case, SLM takes no action to stop a process.
	`child`	`[ self | before | after ]`	For any process launched by SLM, its child processes are out of SLM's direct control. You can specify the shutdown of these child processes as relative to when the SLM-controlled parent process is terminated. The settings are: `self` (the default), `before`, and `after`.

Copyright © 2014, QNX Software Systems Limited

Tag	Attribute	Value(s)	Description
		data	Contains the signal number to send the process to stop it. By default, `SIGTERM` is sent, but you can change this to any signal. If repeated failed attempts to stop the process fail, `SIGKILL` is sent. This tag value isn't needed when the `stop` attribute is set to `none`.
`user`		*uid*`:`*gid*	The user ID and group ID to assign to the underlying process. The two strings are separated by a colon (e.g., `jgarvey:techies`).
`waitfor`	`wait`	`[ none | delay | pathname | exits | blocks ]`	Once a component has been launched, SLM can wait for that component to set itself up before starting any dependent components. Values: • `none` (the default)—causes SLM to start other components immediately • `delay`—SLM pauses for the specified number of milliseconds • `pathname`—SLM probes for the appearance of the specified pathname • `exits`—SLM waits for the process to finish • `blocks`—SLM waits for a specified thread in the process to reach the `RECV-blocked` state
	`polltime`	*poll_time*`:`*timeout_time*	Use with `wait="pathname"` or `wait="exits"` to specify a polling interval and total wait time (both in milliseconds) that override the global values. For example, `polltime="100:20000"` results in polling every 100 milliseconds and timing out after 20 seconds.
		data	Contains data specific to the specified wait condition. For example, for `delay`, the value could be 5000 (for a 5s delay).

Only the `command` property is mandatory—all processes must have a path to the binary executable. The remaining properties are optional.

Copyright © 2014, QNX Software Systems Limited

Modules

You can group components into *modules*. The processes within a module could make up a subsystem or could be used to establish a set of system states, such as a base level of operation and various higher levels. Modules must be named so they can be internally referenced. Each module must be described in a tag, as follows:

```
<SLM:module name="device_monitors">
    -- module description --
</SLM:module>
```

To list the components within a module use the `member` tag. There are no attributes for `member` tags; the tag values refer to member components by the internal names defined in their respective `component` tags.

You can include multiple components in a module by using one `member` tag with wildcards in the component names. For example, you can write:

```
<SLM:member>devb-*</SLM:member>
```

Modules can have dependencies on components or on other modules. Each dependency must be specified in a `depend` tag within the `module` tag.

Components and modules may be specified in any order in the XML configuration file, but SLM will raise an error if any circular dependencies are found.

Sample configuration file

Suppose you want to automate the setup of your system's IP connectivity. This would require running `io-pkt`, which creates an IP socket for network traffic, and then running `ifconfig` to bind an IP address to the socket. You can create a module to include two components that correspond to the two services, and then describe the dependency of `ifconfig` on `io-pkt` in the component entries. The XML file would then look like this:

```
<SLM:system>
    <SLM:component name="io-pkt">
        <SLM:command>/sbin/io-pkt-v6-hc</SLM:command>
        <SLM:args>-ptcpip stacksize=8192</SLM:args>
        <SLM:waitfor wait="pathname">/dev/socket</SLM:waitfor>
    </SLM:component>
    <SLM:component name="ifconfig">
        <SLM:depend>io-pkt</SLM:depend>
        <SLM:command>/sbin/ifconfig</SLM:command>
        <SLM:args>en0 192.168.1.5 up</SLM:args>
        <SLM:waitfor wait="exits"></SLM:waitfor>
    </SLM:component>
    <SLM:module name="net-setup">
        <SLM:member>io-pkt</SLM:member>
        <SLM:member>ifconfig</SLM:member>
    </SLM:module>
</SLM:system>
```


Copyright © 2014, QNX Software Systems Limited

slogger

System logger

Syntax:

```
slogger [-cm] [-f severity] [-l logfile[,size]] [-s size]
        [-U user_name | uid[:gid[,sup_gid]*]]]
        [-u event_id] [-v[v]...]
```

Runs on:

QNX Neutrino

Options:

-c

Open the log file with `O_SYNC` to forcibly commit the logged events to the disk.

-f *severity*

Log only events to the logfile with this severity or higher. This option is useful only with the -l option; `slogger` always saves all events in its internal buffers. The lowest severity is 7 and the highest is 0. The default is 7.

-l *logfile*[,*size*]

Write a copy to *logfile* of each event that has a severity that's numerically less than or equal to that specified by the -f option. If you specify the optional *size* after the filename, `slogger` alternates between two files, *logfile*`0` and *logfile*`1`, as the files reach the given size.

-m

Use `CLOCK_MONOTONIC` instead of the default `CLOCK_REALTIME` as the clock source.

-s *size*

Maintain a circular in-memory log buffer of this size (in kilobytes) to hold system log messages. When the buffer becomes full, the oldest messages are replaced by new ones as they arrive. The default size is 16 KB.

-U *user_name*

-U *uid*[:*gid*[,*sup_gid*]*]]

Copyright © 2014, QNX Software Systems Limited

(QNX Neutrino 6.6 or later) Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

-u *event_id*

Generate a user-string trace event for all messages received. The given event ID must be in the range from `_NTO_TRACE_USERFIRST` through `_NTO_TRACE_USERLAST`, as defined in `<sys/trace.h>`. For more information, see the System Analysis Toolkit *User's Guide*, and *TraceEvent()* in the QNX Neutrino *C Library Reference*.

The default is not to generate trace events.

-v[v]...

Be verbose; more `v` characters cause more verbosity.

Description:

The system logger, `slogger`, is the central manager for logging applications' system messages. It maintains these messages in a circular buffer, allowing them to be read later or in real time. When the buffer fills, new messages replace the oldest ones in the buffer. If you use the -l and -f options, `slogger` also writes the messages to a log file, as described below.

Each system message is assigned a unique major code, minor code and a severity. The severity levels are as follows:

Manifest Name	Value	Description
`_SLOG_SHUTDOWN`	0	Shut down the system NOW (e.g. for OEM use)
`_SLOG_CRITICAL`	1	Unexpected unrecoverable error (e.g. hard disk error)
`_SLOG_ERROR`	2	Unexpected recoverable error (e.g. need to reset a hardware controller)
`_SLOG_WARNING`	3	Expected error (e.g. parity error on a serial port)

Copyright © 2014, QNX Software Systems Limited

Manifest Name	Value	Description
_SLOG_NOTICE	4	Warning (e.g. out of paper)
_SLOG_INFO	5	Information (e.g. printing page 3)
_SLOG_DEBUG1	6	Debug messages (e.g. normal detail)
_SLOG_DEBUG2	7	Debug messages (e.g. fine detail)

If you specify the -l *logfile* option, `slogger` creates a separate thread that saves to the specified file each system message that's of the severity given in the -f option.

If you use the optional *size* argument to the -l option, `slogger` alternates between two files, *logfile*`0` and *logfile*`1`. When one file reaches the size given, `slogger` closes it and opens the other file. This file is truncated to zero length and new messages are written to it until it fills up. The process of alternating then repeats. This way you always have one file containing the latest messages and another file containing the most recent history.

Applications that can't assume that they have a standard console output that's viewable should use `slogger` for logging notices, warnings and errors. This includes all drivers and resource managers.

The `slogger` manager creates these devices in the pathname space:

`/dev/slog`

Used to read and write system log messages.

`/dev/console`

Implements a simple console tty that saves what you write to it in the system log. Reads always return end-of-file.

Applications wishing to post a new system log message should use the library routines *slogb()*, *slogf()*, *slogi()*, and *vslogf()*, which properly format the data and issue a write to `/dev/slog`.

To view the log, use *`sloginfo`* (p. 1818). Applications wishing to read system log messages may open `/dev/slog` for reading and issue *read()* calls. Each read may return one or more messages, but messages are never split across a read.

More than one application may open `/dev/slog` for reading at a time. Each sees its own copy of the data and doesn't affect the others. This allows multiple filters, each

Copyright © 2014, QNX Software Systems Limited

looking for certain log messages, to be run in parallel. Here's a simple filter that prints codes and severities:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <sys/slog.h>

int main(int argc, char *argv[]) {
    int events[4096];
    int *evp;
    int n;
    int wait = 0;  // Set to 1 to block and print
                   // new events as they arrive.
    int fd;

    fd = open("/dev/slog",
              wait ? O_RDONLY : O_RDONLY|O_NONBLOCK);

    for(;;) {
        int cnt;

        // Read some events.
        n = read(fd, events, sizeof(events));
        if(n == -1) {
            // Normal case for a non-blocking read
            // with no events.
            if(errno == EAGAIN) {
                exit(EXIT_SUCCESS);
            }

            exit(EXIT_FAILURE);
        }

        // Converts bytes to ints (all events are
        // composed of ints).
        n /= sizeof(int);
        if(n == 0)
            break;

        for(evp = events ; evp < &events[n] ;
            evp += cnt) {
            int    major, minor, severity, txt;
            time_t sec;
            char   timebuf[60];

            major    = _SLOG_GETMAJOR(evp[1]);
            minor    = _SLOG_GETMINOR(evp[1]);
            cnt      = _SLOG_GETCOUNT(evp[0]) +
                       _SLOG_HDRINTS;
            severity = _SLOG_GETSEVERITY(evp[0]);
            txt      = _SLOG_GETTEXT(evp[0]);

            sec = evp[2];
            strftime(timebuf, sizeof(timebuf),
                     "%h %d %T", localtime(&sec));
            printf("%s    %d %5d    %2d ", timebuf,
                   severity, major, minor);

            if(txt)
                printf("%s",
                       (char *) &evp[_SLOG_HDRINTS]);
        }

    }

    exit(EXIT_SUCCESS);
}
```

You can clear the system log buffer by calling `unlink("/dev/slog")` from a program, or `rm /dev/slog` from the shell.

Copyright © 2014, QNX Software Systems Limited

The `slogger` manager replaces the older Unix system logger daemon, *`syslogd`* (p. 1885). The Unix library routines for `syslogd` output their messages to `/dev/console`, which is caught and logged by `slogger`.

You should start `slogger` as soon as possible after the system boots. Otherwise, messages from managers or drivers that were started before it may be lost. If `slogger` is killed (or never started), all messages that programs send to `slogger` are lost. If `slogger` is started or restarted, new messages go to the new `slogger`.

Examples:

Log system messages:

```
slogger
```

Maintain an awesome in-memory buffer for system messages:

```
slogger -s 100k
```

In addition to keeping an in-memory buffer, save in a file those system log messages of severity 0, 1, or 2:

```
slogger -l /var/logs/slogs -f 2
```

In addition to keeping an in-memory buffer, save all system log messages in a file. Since a size option is specified, the log file alternates between `slogs0` and `slogs1`, switching as each file reaches a size of 100 KB:

```
slogger -l /var/logs/slogs,100k
```

Copyright © 2014, QNX Software Systems Limited

slogger2

System logger

Syntax:

```
slogger2 [-b buffer_set[,buffer_set...]] [-c] [-p pid]
         [-U user_name | uid[:gid[,sup_gid]*]]]
         [-v[v]...]
```

Runs on:

QNX Neutrino

Targets:

ARM, x86

Options:

-b *buffer_set*[,*buffer_set*...]

Specify the buffer sets to clear with the -c option.

-c

Clear the log buffers. If you specify the -b, only those buffers are cleared; otherwise, all buffers are cleared.

-p *pid*

(Debugging only) Run as the given client process ID.

-U *user_name*

-U *uid*[:*gid*[,*sup_gid*]*]]

Once running, run as the specified user, so that the program doesn't need to run as `root`:

- In the first form, the service sets itself to be the named user and uses that user's groups. This form depends on the `/etc/passwd` and `/etc/group` files.
- In the second form, the service sets its user ID, and optionally its group ID and supplementary groups, to the values provided.

-v[v]...

Copyright © 2014, QNX Software Systems Limited

Be verbose; more v characters cause more verbosity.

Description:

The slogger2 daemon is the central manager of a system-logging framework that overcomes various limitations of the legacy slogger system. The primary goals for slogger2 are to:

- provide high performance
- work in situations where interrupts aren't enabled
- provide component-specific log buffers and verbosity control

The components of the system logger include:

libslog2, <slog2.h>

The library and header file for the API.

slogger2

The daemon process that's responsible for:

- allocating buffers for each process
- maintaining buffers (cleaning up buffers when processes die, etc.)
- controlling verbosity
- gathering logs
- providing a means for other components to consume the logs

libslog2parse

A library that supports log parsing by other tools (e.g., slog2info).

slog2info

The realtime viewer. It can consume the live stream from slogger2, as well as parse (using libslog2parse) and display saved slog2 files.

Buffer Management

The central slogger2 process manages a region of shared RAM where all slog2 buffers are located. This gives slog2 the performance benefit of writing to RAM (instead of to flash or disk) without losing logs in the case of process crashes. However, if the power source is disconnected, any log contents in RAM will be lost.

Individual slog2 instances are allocated from this memory region by slogger2 and are provided to the client process, via *slog2_register()*, in the form of a shared memory file. The libslog2 library uses *shmem_open()* to access each instance.

Within each slog2 instance there may be up to four buffers defined by the client process. These buffers can be any combination of 4 KB pages. The primary intent for

Copyright © 2014, QNX Software Systems Limited

these buffers is to enable high-rate logging into one buffer, while preserving a longer history of low-rate logging in another.

The intent isn't for every subcomponent in a client process to have its own buffer. Each buffer introduces overhead, both in memory management and in terms of log-streaming performance. The expectation is that it will be rare for more than two buffers in a single instance to be used. For extreme situations, the client process can register multiple `slog2` instances to satisfy a need for additional buffers.

The central `slogger2` process is responsible for cleaning up the `slog2` instances, including calling the *shmem_unlink()* function to remove the shared memory file.

Severity, verbosity, and filtering

Each log line is assigned one of the following severity levels (listed here in decreasing order):

`SLOG2_SHUTDOWN`

Shut down the system **now** (e.g., for OEM use).

`SLOG2_CRITICAL`

Unexpected unrecoverable error (e.g., hard disk error).

`SLOG2_ERROR`

Unexpected recoverable error (e.g., you need to reset a hardware controller).

`SLOG2_WARNING`

Expected error (e.g., parity error on a serial port).

`SLOG2_NOTICE`

Warning (e.g., out of paper).

`SLOG2_INFO`

Information (e.g., printing page 3).

`SLOG2_DEBUG1`

Debug messages (e.g., normal detail).

`SLOG2_DEBUG2`

Debug messages (e.g., fine detail).

The verbosity level controls which log lines are *written* in the `slog2` buffer; if the severity level is greater than the verbosity level, the line is written in the buffer.

Copyright © 2014, QNX Software Systems Limited

Filtering controls which log items are *displayed* to the user; the log contents aren't affected. You could filter the log by using (for example) `grep`, `slog2info`, or a custom log viewer.

Copyright © 2014, QNX Software Systems Limited

slog2info

Display messages from the system log

Syntax:

```
slog2info [options]
```

Runs on:

QNX Neutrino

Options:

-b *bset1*,*bset2*,...

Parse only the logs of the given buffer set names.

-c

Clear the live `slog2` buffers. You can use this option with -b to clear specific buffer sets.

-h

Display a usage message.

-i

Display information about a buffer set specified by the -l option.

-l *filename*

("el") The buffer set file to parse. The *filename* argument must be the full path of a file under `/dev/shmem/slogger2/`.

-r [*resetfilename*]

Display logs preserved through the most recent reset. Optionally specify a reset file to display logs from another reset.

-w

Wait mode; listen for new prints.

Description:

The `slog2info` utility displays the contents of the system log buffer managed by *`slogger2`* (p. 1812), which must be running to record these messages.

Copyright © 2014, QNX Software Systems Limited

If you don't specify any options, `slog2info` displays all the logs.

Examples:

Dump all logs for a live system:

```
slog2info
```

Dump the logs only from a given buffer_set file:

```
slog2info -l /dev/shmem/slogger2/my_test_buff.86749364
```

Dump the logs from all buffer_sets with a matching name:

```
slog2info -b my_test_buff
```

Dump all logs from the latest reset:

```
slog2info -r
```

Dump the logs from all buffer_sets with a matching name from the latest reset:

```
slog2info -r -b my_test_buff
```

Dump information pertaining to a given buffer set file:

```
slog2info -i -l /dev/shmem/slogger2/my_test_buff.86749364
```

Dump all logs from a live system, and then wait:

```
slog2info -w
```

Copyright © 2014, QNX Software Systems Limited

sloginfo

Print messages from the system log

Syntax:

```
sloginfo [options] [filename]
```

Runs on:

QNX Neutrino

Options:

-c

Clear the log buffer after displaying all waiting events.

-h

Print unformatted entries in hexadecimal. By default, they're printed in decimal.

-m *code*

Display events with this major code (default: display all).

-s 0..7

Display events with this severity or lower (default: 7). The lowest severity is 7 and the highest is 0.

-t

Print *time* for events with millisecond resolution.

-w

Wait for more events to arrive.

filename

The name of the file containing raw events (default: `/dev/slog`).

Description:

The `sloginfo` utility prints the contents of the system log buffer managed by *`slogger`* (p. 1807), which must be running to record these messages.

Copyright © 2014, QNX Software Systems Limited

By default, `sloginfo` prints all messages and exits. You can use the -w option to have it wait for more messages.

You can use the -m and -s options to filter the messages to print. There are 16 million major codes, with 4096 minor codes for each major code. Selecting a major code displays messages with any of the minor codes for the specified major code. You can use the major codes to display messages for a specific area, such as TCP/IP. The major codes are defined in the file `/usr/include/sys/slogcodes.h`.

If a filename is specified, `sloginfo` takes its input from that file instead of `/dev/slog`. It's assumed that the file contains raw system log data saved by a program or is the file specified in the -l option to `slogger`.

Examples:

Print out all system log messages and exit:

```
sloginfo
```

Print out all system log messages and wait for more to arrive, printing them as they do:

```
sloginfo -w
```

Print out all system log messages, clear the log, and exit:

```
sloginfo -c
```

Display only system log messages with a severity of 0, 1, 2, 3, or 4:

```
sloginfo -s 4
```

Display the system log messages stored in a specific file:

```
sloginfo /var/logs/slog0
```

Files:

`/dev/slog`

The default system log file.

Copyright © 2014, QNX Software Systems Limited

/etc/socks.conf

SOCKS clients configuration file

Name:

/etc/socks.conf

Description:

All SOCKS client programs use this file to:

- determine whether to use direct or proxy connection to a given destination host
- exert access control based on the destination host, the requested service (port number on the destination host), and the effective userid of the requesting local user.

Each line in the file may be up to 1024 characters long. Lines starting with a number sign (#) are comments. Lines that aren't comments must be of one of the three forms:

```
deny [*=userlist] dst_addr dst_mask [op dst_port] [: shell_cmd]
direct [*=userlist] dst_addr dst_mask [op dst_port] [: shell_cmd]
sockd [@=serverlist] [*=userlist] dst_addr dst_mask [op dst_port] [: shell_cmd]
```

A deny line tells the SOCKS clients when to reject a request.

A direct line tells when to use a direct connection.

A sockd line tells when to use a proxy connection and, optionally, which SOCKS proxy server or servers to try.

Spaces and tabs separate the fields. Fields enclosed in square brackets are optional. The fields are:

userlist

One or more userids or filenames, separated by commas. No spaces or tabs are allowed in the list. The userids should be of users on the local host, not those on the destination host or the SOCKS server host.

The filenames must be full pathnames with the leading /. Inside the specified files, userids may be listed one or several per line, with any combination of blanks, tabs, and commas as separators.

A # at the beginning of a line marks the remainder of the line as a comment. Each line in these files may be up to 1023 characters long. If the *= *userlist* field is omitted, the line applies to all userids.

dst_addr dst_mask

Copyright © 2014, QNX Software Systems Limited

These fields together specify the destination IP address or the range of destination IP addresses. Both are given in the usual dotted form (e.g. `129.1.2.3`).

Bits in *dst_mask* that are set to 0 indicate the bit positions to be masked off (i.e. ignored) during comparison of *dst_addr* and the actual destination IP address.

For example, specifying `255.255.255.255` in *dst_mask* demands an exact match with *dst_addr*, whereas `0.0.0.0` is an address match no matter what is specified for *dst_addr*. (Note that this is the same way netmasks are interpreted, but is the direct opposite of how the address masks are used in Cisco router access lists.)

op

One of the following:

`eq`

Equal.

`neq`

Not equal.

`lt`

Less than.

`gt`

Greater than.

`le`

Less than or equal.

`ge`

Greater than or equal.

dst_port

Either a port number (e.g. 23) or the equivalent service name as specified in the *`/etc/services`* (p. 1744) file (e.g. `telnet` for port number 23). If this pair is omitted, the line applies to all services.

serverlist

Copyright © 2014, QNX Software Systems Limited

Used only in a `sockd` line. It consists of one or more SOCKS proxy servers that the client program should try to use (in the indicated order) for establishing a proxy connection.

You can use only commas as separator — no spaces or tabs are allowed in the list. Although domain names of the servers may be used in the list, it's probably more prudent to specify IP addresses.

If the *serverlist* field is omitted, the client program uses the default SOCKS proxy server, which is determined by the environment variable ***SOCKS_SERVER*** or by the name compiled into the SOCKS client program.

Consider this `sockd` line:

```
sockd @=1.2.3.4 *=boss,root 11.12.13.14 255.255.255.255 eq telnet
```

To match the condition indicated in this line, a request must come from a local user whose effective id is either `boss` or `root`, the destination IP address must be `11.12.13.14` exactly, and the service requested must be `telnet`. In that case, connection to host `11.12.13.14` should be done via a SOCKS proxy server on host `1.2.3.4`.

Every time a SOCKS client has to make a network connection, it checks the pending request against the `/etc/socks.conf` file, one line at a time. Once the client finds a line with conditions that are matched by the request, the action specified on that line is taken. The remaining lines of `/etc/socks.conf` are skipped. If no matching line is found throughout the file, the request is denied.

The order of the lines in the file is extremely important — switch two lines and you may have entirely different results!

shell_cmd

A command string that's executed when the conditions on that line are satisfied. The following substitutions occur before the string is presented to the Borne shell for execution:

These characters:	Are replaced by:
%A	The client host's domain name if known; by its IP address otherwise
%a	The client host's IP address
%c	“connect” or “bind”
%p	The client program's process ID

Copyright © 2014, QNX Software Systems Limited

These characters:	Are replaced by:
%S	The service name (e.g. `ftp`) if known; by the destination port number otherwise
%s	The destination port number
%U	The userid at login
%u	The effective userid
%Z	The destination host's domain name if known; by its IP address otherwise
%z	The destination host's IP address
%%	A single %

You can string several shell commands together in the usual way with `|`, `;`, etc.

Although there's an implied "deny all" at the end of the control file, you may supply one explicitly to take some specific action when requests are so rejected. For example:

```
deny 0.0.0.0 0.0.0.0 : /usr/bin/mail -s `SOCKS: rejected %S from %u to %Z' root
```

Unlike the previous version, connection to address `127.0.0.1` (localhost) is *always* done directly, so there's no need to specify that in `/etc/socks.conf`.

Environment variables:

SOCKS_SERVER

If defined, this environment variable specifies the name or IP address of the SOCKS proxy server host to use, overriding the default server compiled into the programs.

Copyright © 2014, QNX Software Systems Limited

sockstat

List the open sockets

Syntax:

```
sockstat [-46clnu] [-f address_family] [-p ports]
```

Runs on:

QNX Neutrino

Options:

-4

Show AF_INET (IPv4) sockets.

-6

Show AF_INET6 (IPv6) sockets.

-c

Show connected sockets.

-f *address_family*

Limit the listed sockets to those of the specified *address_family*. The following address families are recognized: `inet` for `AF_INET`; `inet6` for `AF_INET6`; and `local` or `unix` for `AF_LOCAL`.

-l

Show the listening sockets.

-n

Numeric output only. No attempt is made to look up symbolic names for addresses and ports.

-p *ports*

Show only Internet sockets if either the local or foreign port number is in the specified list. The *ports* argument is a comma-separated list of port numbers and ranges specified as a first and a last port separated by a dash.

-u

Show `AF_LOCAL` (UNIX) sockets.

Copyright © 2014, QNX Software Systems Limited

Description:

The `sockstat` command lists open Internet or UNIX domain sockets.

If you don't specify any of the -4, -6, or -u options, `sockstat` lists the sockets in all three domains.

If you don't specify either of the -c or -l, `sockstat` lists both listening and connected sockets, as well as those sockets that are in neither state.

The information listed for each socket is:

USER

The user who owns the socket.

COMMAND

The command that holds the socket.

PID

The process ID of the command that holds the socket.

FD

The file descriptor number of the socket.

PROTO

The transport protocol associated with the socket for Internet sockets, or the type of socket (stream or datagram) for UNIX sockets.

LOCAL ADDRESS

For Internet sockets, this is the address to which the local end of the socket is bound (see *getsockname()* in the QNX Neutrino *C Library Reference*). For bound UNIX sockets, it's the socket's filename or `-`.

FOREIGN ADDRESS

The address to which the foreign end of the socket is bound (see *getpeername()*), or `-` for unconnected UNIX sockets.

Copyright © 2014, QNX Software Systems Limited

sort

Sort, merge, or sequence-check text files (POSIX)

Syntax:

```
sort [-m] [-o name] [-bdfinru] [-t char]
     [-k keydef] [+oldkey] [file...]

sort [-c] [-bdfinru] [-t char] [-k keydef]
     [+oldkey] [file]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-c

Check that the single input file is already sorted. Produce no output.

-m

Merge only. It's assumed that the input files are already sorted.

-o *name*

The *name* argument is the name of an output file to be used instead of the default, which is standard output. This file can be the same as one of the input files.

-u

Unique; suppress all but one key in each set of lines having equal keys.

The following options override the default ordering rules. When ordering options appear independent of key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see option -k), the specified ordering options override all global ordering options for that key.

-b

Ignore leading blank characters in field comparisons.

-d

Sort in “dictionary order.” Only blank characters and alphanumeric characters are significant in comparisons.

Copyright © 2014, QNX Software Systems Limited

-f

Fold uppercase letters into lowercase.

-i

Ignore all nonprintable characters.

-k *keydef*

Define *keydef* to be a sort key.

-n

Interpret the field as numeric, including the sign and optional “thousands” separator. Sort in numeric order. This implies the -b option.

-r

Reverse the sort order.

-t *char*

Use *char* as the field separator character.

file

The pathname of a file to be sorted, merged, or checked. If you don't specify any files to be sorted, or if a *file* operand is the dash character (-), the standard input is used.

+oldkey

This is a historical key-field mechanism. For more information, see below.

Description:

The `sort` utility orders lines from a set of files (or standard input if no files are provided) and merges the output into the specified output (or standard output if no output file is specified). The `sort` utility provides a number of options for controlling the sorting mechanism. The default behavior of `sort` is to treat all lines as a sequence of characters to be sorted in ascending order.

The `sort` utility regards files and file contents the following way:

- a file is a sequence of lines
- a line is a sequence of fields followed by a newline
- a field is a sequence of characters followed by a field separator
- a field separator, by default, is a blank, although this may be altered by the -t option

Copyright © 2014, QNX Software Systems Limited

The -t option has subtle effects. If you *don't* use the option, only the first blank in a string of blank characters is interpreted as a field separator; the remaining are considered part of the next field. If you use the -t option, every occurrence of the field separator character is considered a terminator for a field. Thus, four adjacent field separators represent three empty fields.

A sort key field may be defined using the following syntax:

```
-k field_start [type_string] [,field_end] [type_string]
```

where *field_start* and *field_end* specify the beginning and end of the key field, and *type_string* is used to specify attributes specific to the key.

The *field_start* and *field_end* are each specified by a pair of digits of the form *m.n*, where the *m* refers to the field starting after the *m*th field separator in a line. For *field_start*, the *.n* refers to the *n*th character of the specified field, and is taken as zero if not specified. For *field_end*, the *.n* refers to the *n*th character after the last character of the specified field, and is taken as zero if not specified.

The *type_string* may be formed from the characters `bdfinr`, which apply their defined attributes to the determination of the key.

There is another mechanism (*+oldkey*) for defining sort keys, which remains for historical reasons. The syntax of this mechanism is:

```
+field_start [type_string] -field_end [type_string]
```

The semantics are the same as for the -k option.

The following command sorts the password database by *gid*, then *uid*, then *username*:

```
sort -t: -k 3.0n -k 2.0n -k 0.0d /etc/passwd
```

Notice that the *type_string* attached to *field_string* is used to determine the type of comparison with this key.

The second *type_string* is solely to adjust the extent of the *sort_key*, thus only `i` and `b` make any sense. Any other characters are ignored.

Examples:

The standard password file has the form:

```
username:password:uid:gid:misc:home:shell
```

Sort the password database by *username*:

```
sort -t: /etc/passwd
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

All input files were sorted successfully, or the option -c was specified and the input file was already correctly sorted.

1

The -c option was used and the input file wasn't already sorted.

> 1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

spatch

Fullscreen patch utility (QNX Neutrino)

Syntax:

```
spatch [-bp] file [offset]
```

Runs on:

QNX Neutrino

Options:

-b

Browse only; don't allow the Save command. The file or disk is accessed in read-only mode.

-p

Pause before starting, for example to allow a floppy disk to be inserted in a disk drive.

file

The file or disk to be examined.

offset

The address — in RAM, in the file, or on the disk — where the spatch is to begin (specified in hex).

Description:

The spatch utility provides fullscreen editing of files or disk blocks. The screen displays a 16-by-16 (256) byte image of the data being examined, similar to that shown here:

```
 Edit Next Prev Lastblk Home Goto Find Continue Save Addr Quit
000000000: 2E 28 6E 65 77 29 20 53  50 41 54 43 48 20 22 46 .(new) SPATCH "F
000000010: 75 6C 6C 20 73 63 72 65  65 6E 20 70 61 74 63 68 ull screen patch
000000020: 20 75 74 69 6C 69 74 79  22 1E 2E 28 73 79 6E 74  utility"..(synt
000000030: 61 78 29 1E 09 11 73 70  61 74 63 68 10 20 20 11 ax)...spatch.  .

000000040: 66 69 6C 65 10 20 20 AE  66 69 6C 65 6E 61 6D 65 file.  .filename
000000050: AF 1E 09 11 73 70 61 74  63 68 10 20 20 11 64 69 ....spatch.  .di
000000060: 73 6B 10 20 20 AE 64 72  69 76 65 AF 20 20 AE 62 sk.  .drive.  .b
000000070: 6C 6F 63 6B AF 1E 09 11  73 70 61 74 63 68 10 20 lock....spatch.

000000080: 20 11 6D 65 6D 10 20 20  AE 73 65 67 6D 65 6E 74  .mem.  .segment
000000090: AF 20 20 AE 6F 66 66 73  65 74 AF 1E 2E 28 65 78 .  .offset...(ex
0000000a0: 61 6D 70 6C 65 73 29 1E  09 11 73 70 61 74 63 68 amples)...spatch
0000000b0: 20 20 66 69 6C 65 20 20  2F 63 6D 64 73 2F 6C 73   file   /bin/ls

0000000c0: 1E 09 73 70 61 74 63 68  20 20 64 69 73 6B 20 20 ..spatch  disk
0000000d0: 31 20 20 31 1E 09 73 70  61 74 63 68 20 20 6D 65 1  1..spatch  me
0000000e0: 6D 20 20 62 30 30 30 20  20 30 10 1E 2E 28 73 74 m  b000  0...(st
0000000f0: 61 72 74 29 1E 53 50 41  54 43 48 20 69 73 20 61 art).SPATCH is a
```


Copyright © 2014, QNX Software Systems Limited

At the top of the screen, there's a list of commands. To select a command, either type its first letter or move the cursor to the command (with the arrow keys) and press **Enter**.

The commands are as follows:

Edit

Enter the data area. Pressing **Tab** switches between hex and ASCII data entry. Pressing **Esc** returns you to the menu. The changed data isn't updated on the disk or in memory.

Next

Move forward 256 bytes. You can also press **Pg Dn**.

Prev

Move backward 256 bytes. You can also press **Pg Up**.

Home

Go to the start of the file, disk, or memory. You can also press the **Home** key.

Lastblk

Go to the last block of the file or disk. You can also press **End**.

Goto

Move to a specified address. The type of address depends on the source of the data (file or disk) and the address type.

Find

Search for a specified pattern, which may consist of single characters or hex digit pairs separated by a space. For example, the patterns `61 62 63 d e` and `a b c d e` both match the five characters `abcde`.

To stop the search, press any key.

Continue

Find the next occurrence of the last pattern found. You typically use this command after a `Find` when searching.

Save

Save the current screen back to the source. Without issuing this command, all changes made using `Edit` are lost as soon as you leave the current screen of data. The `Save` option is disabled if you specify the -b (browse) option.

Addr

Copyright © 2014, QNX Software Systems Limited

Toggle between address types:

- Absolute (default for FILE)
- Disk Block:Offset (default for DISK)

Quit

Leave the `spatch` utility.

If you specify a directory in the *file* argument, the disk is edited, but `spatch` moves only through the blocks that make up the directory. This is similar to "spatching" a file, but if you wish to make changes, the disk must be opened for exclusive use as would any block special file.

To run `spatch` on a directory or a block special file, you must either be `root` or have write permission for the disk's block special file.

The *offset* argument lets you specify the address where `spatch` is to begin. If *file* is a regular file, the offset is a byte offset. If *file* is a block special file, the offset is a block:byte offset. If *file* is a block special file with a QNX filesystem on it, the offset may be the name of a file or directory (the beginning address is the first block of the named file). If the named file has extents, `spatch` doesn't thread through those extents, but goes to the next sequential block.

You can use `spatch` to recover lost files or directories. For more information, see the Working with Filesystems and Backing Up and Recovering Data chapters of the QNX Neutrino *User's Guide*.

Examples:

Patch the contents of the file `/bin/ls`:

```
spatch /bin/ls
```

Patch the contents of the block-special raw disk volume `/dev/hd0`:

```
spatch /dev/hd0
```

Environment variables:

TERM

Interpreted as the name of the terminal type.

Exit status:

0

Success.

Copyright © 2014, QNX Software Systems Limited

> 0

An error occurred.

Caveats:

You may not use the `spatch` utility on a disk when there are open files on the disk, unless you specify the -b (browse) option.

Copyright © 2014, QNX Software Systems Limited

split

Split files into pieces (POSIX)

Syntax:

```
split [-a suffix_length] [-b n[k|m]] [-l line_count]
      [-p pattern] [file [name]]
```

Runs on:

QNX Neutrino

Options:

-a *suffix_length*

Use *suffix_length* letters to form the suffix of the filenames of the split file.

-b *n*[klm]

Split a file into pieces that are *n* bytes in size. You can add the letter k or m after *n* to specify units of kilobytes (1024 bytes) or megabytes (1048576 bytes).

-l *line_count*

("el")Split the files so that there are *line_count* lines in each resulting file piece. The *line_count* argument is an unsigned decimal integer. The default is 1000. Note that the last file might differ in size from the other files.

-p *pattern*

(non-POSIX extension) Split the file whenever an input line matches *pattern*, which is interpreted as an extended regular expression. The matching line will be the first line of the next output file. Note that this option is incompatible with the -b and -l options.

file

The pathname of the file to be split. If you don't specify any files, or *file* is -, `split` reads from standard input.

name

The prefix to be used for each of the files resulting from the split operation. If you don't specify a *name*, x is used as the prefix of the output files. The combined length of *name* and *suffix_length* can't exceed 48 characters.

Copyright © 2014, QNX Software Systems Limited

Description:

The `split` utility reads an input file and writes the data from that file into one or more output files.

By default, the names of the output files are `xaa`, `xab`, ..., `xzz`, and each output file, except possibly the last, gets 1000 lines.

The last file contains the remainder of the input file, and therefore may be smaller than the requested size. Conversely, it may be longer than the other files if there are too few filenames available to take all the input in chunks of the specified size.

Examples:

Suppose you have a file named `big_file` that's 8192 lines in length. The following command creates nine files named `xaa`, `xab`, `xac`, ..., `xai`. The first eight files all contain 1000 lines, while the last file contains only 192:

```
split big_file
```

Again, assuming that `big_file` is 8192 lines in length, the following command creates only two files: `smaller_aa`, which contains 8000 lines, and `smaller_ab`, which contains 192 lines:

```
split -l 8000 big_file smaller_
```

Files:

You can use any file as input, but if you're splitting a nontext file, you must specify option -b. The output files contain portions of the original input file that are otherwise unchanged.

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

spooler

Handle requests for a printer's resources

You must be `root` to start `spooler`.

Syntax:

```
spooler -d device [options]
```

Runs on:

QNX Neutrino

Options:

-C

Calculate and print the device ID, and then exit.

-c *config*

The name of the configuration file that defines filters and properties for a printer.

-d *device*

The output device (default: `/dev/null`).

-F

Disable file spooling and connect applications to filters using FIFOs.

-f *maxjobs*

Limit the number of simultaneous spool files to *maxjobs*. The default is no limit.

-g

Make the printer a global network resource.

-n *name*

The name of the printer. If this option isn't specified, `spooler` gets the name from the PnP model or the configuration file.

-P *PnPstr*

Copyright © 2014, QNX Software Systems Limited

The PnP string for a printer (default: query printer).

-s *spooldir*

The spool directory (default: `/var/spool/printers/`*printer-name*).

-v[v...]

Verbose output; each addition `v` increases verbosity.

Description:

The `spooler` allows multiple users to share the resources of a single printer. If you have more than one printer, you can start an instance of `spooler` for each one.

You can start `spooler` (e.g. for a network printer) in your `/etc/rc.d/rc.local` file.

When you start `spooler`, it queries the printer to determine its type. If you provide a PnP string, `spooler` doesn't query the printer for it:

```
spooler -d /dev/par1 -P PnPstring
```

You can use other options to force `spooler` to use a configuration file and assign a name to the printer, but this isn't recommended:

```
spooler -c pcl -n My_PCL_Printer -d /dev/par1 &
```

You can find configuration files for commonly used printers in `/etc/printers`. These files specify the possible and default settings for the printers, as well as filters for converting output to a form that the printers understand.

Copyright © 2014, QNX Software Systems Limited

ssh

OpenSSH SSH client: remote login program

Syntax:

```
ssh [-1246AaCfgKkMNnqsTtVvXxY] [-b bind_address]
    [-c cipher_spec] [-D [bind_address:]port]
    [-e escape_char] [-F configfile] [-i identity_file]
    [-L  [bind_address:]port:host:hostport]
    [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option]
    [-p port] [-R [bind_address:]port:host:hostport]
    [-S ctl_path] [-w local_tun[:remote_tun]]
    [user@]hostname [command]
```

Runs on:

QNX Neutrino

Options:

See *ssh* in the NetBSD documentation.

Description:

The ssh (SSH client) utility is a program for logging into a remote machine and for executing commands on a remote machine. For more information, see *ssh* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

ssh-add

Add RSA or DSA identities to the authentication agent

Syntax:

```
ssh-add [-cDdLlXx] [-t life] [file ...]
ssh-add -s reader
ssh-add -e reader
```

Runs on:

QNX Neutrino

Options:

See *ssh-add* in the NetBSD documentation.

Description:

The `ssh-add` adds RSA or DSA identities to the authentication agent, `ssh-agent`. For more information, see *ssh-add* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

ssh-agent

Authentication agent

Syntax:

```
ssh-agent [-c | -s] [-d] [-a bind_address] [-t life]
          [command [arg ...]]

ssh-agent [-c | -s] -k
```

Runs on:

QNX Neutrino

Options:

See *ssh-agent* in the NetBSD documentation.

Description:

The `ssh-agent` is a program that holds private keys used for public key authentication (RSA, DSA). For more information, see *ssh-agent* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

~/.ssh/ssh_config, /etc/ssh/ssh_config

OpenSSH SSH client configuration files

Name:

~/.ssh/ssh_config, /etc/ssh/ssh_config

Description:

The ssh program obtains configuration data from the following sources in the following order:

1. command-line options
2. the user's configuration file (~/.ssh/config)
3. the system-wide configuration file (/etc/ssh/ssh_config)

For more information, see *ssh_config* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

ssh-keygen

Authentication key generation, management, and conversion

Syntax:

```
ssh-keygen [-q] [-b bits] -t type [-N new_passphrase]
           [-C comment] [-f output_keyfile]

ssh-keygen -p [-P old_passphrase] [-N new_passphrase]
           [-f keyfile]

ssh-keygen -i [-f input_keyfile]

ssh-keygen -e [-f input_keyfile]

ssh-keygen -y [-f input_keyfile]

ssh-keygen -c [-P passphrase] [-C comment] [-f keyfile]

ssh-keygen -l [-f input_keyfile]

ssh-keygen -B [-f input_keyfile]

ssh-keygen -D reader

ssh-keygen -F hostname [-f known_hosts_file]

ssh-keygen -H [-f known_hosts_file]

ssh-keygen -R hostname [-f known_hosts_file]

ssh-keygen -U reader [-f input_keyfile]

ssh-keygen -r hostname [-f input_keyfile] [-g]

ssh-keygen -G output_file [-v] [-b bits] [-M memory]
           [-S start_point]

ssh-keygen -T output_file -f input_file [-v] [-a num_trials]
           [-W generator]
```

Runs on:

QNX Neutrino

Options:

See *ssh-keygen* in the NetBSD documentation.

Description:

The `ssh-keygen` utility generates, manages, and converts authentication keys for `ssh`. For more information, see *ssh-keygen* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

ssh-keyscan

Gather SSH public keys

Syntax:

```
ssh-keyscan [-46Hv] [-f file] [-p port] [-T timeout]
            [-t type] [host | addrlist namelist] [...]
```

Runs on:

QNX Neutrino

Options:

See *ssh-keyscan* in the NetBSD documentation.

Description:

The `ssh-keyscan` utility gathers the public SSH host keys of a number of hosts. For more information, see *ssh-keyscan* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

ssh-keysign

SSH helper program for host-based authentication

Syntax:

```
ssh-keysign
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `ssh-keysign` program is used by `ssh` to access the local host keys and generate the digital signature required during host-based authentication with SSH protocol version 2. For more information, see *ssh-keysign* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

sshd

OpenSSH SSH daemon

Syntax:

```
sshd [-46Ddeiqt] [-b bits] [-f config_file]
     [-g login_grace_time] [-h host_key_file]
     [-k key_gen_time] [-o option] [-p port]
     [-u len]
```

Runs on:

QNX Neutrino

Options:

See *sshd* in the NetBSD documentation.

Description:

The sshd (OpenSSH Daemon) is the daemon program for ssh. Together, these programs replace *rlogin* (p. 1669) and *rsh* (p. 1703), and provide secure encrypted communications between two untrusted hosts over an insecure network. For more information, see *sshd* in the NetBSD documentation.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

/etc/ssh/sshd_config

OpenSSH SSH daemon configuration file

Name:

/etc/ssh/sshd_config

Description:

The sshd daemon reads configuration data from /etc/ssh/sshd_config (or the file specified with the -f option). For more information, see */etc/ssh/sshd_config* in the NetBSD documentation.

Copyright © 2014, QNX Software Systems Limited

`startup-*` options

Generic options for startup programs (QNX Neutrino)

Description:

All QNX Neutrino startup programs support the generic options described below. There are additional options for the following architectures:

- *ARM* (p. 1852)
- *x86* (p. 1852)

Individual startup programs can override these options and may support additional board-specific options. The order of precedence is as follows:

1. board-specific options
2. architecture-specific options
3. generic options

Generic options

-A

Reboot the system on any abnormal termination of the kernel. The default is to display information about the crash, and then halt.

-C

Clear (i.e., zero) any memory allocated by the startup library.

-D *channel*[*.channel_opts*]

Specify an output channel for debugging information. The format of this option and the default value vary from board to board.

-F [~]*value*

Control the *flags* field in the *cpuinfo* section of the system page:

- *value* — OR the *flags* field with *value*
- *~value* — AND the *flags* field with *~value*

For more information about the *flags*, see "Structure of the system page" in the Customizing Image Startup Programs chapter of the *Building Embedded Systems* guide.

-f [*cpu_freq*][,[*cycles_freq*][,*timer_freq*]]

Copyright © 2014, QNX Software Systems Limited

Specify CPU frequencies. All frequencies can be followed by H for hertz, K for kilohertz, or M for megahertz (these suffixes aren't case-sensitive). If no suffix is given, the library assumes megahertz if the number is less than 1000; otherwise it assumes hertz.

If they're specified, *cpu_freq*, *cycles_freq*, and *timer_freq* are used to set the corresponding variables in the startup code:

- *cpu_freq* — the CPU clock frequency. It's also used to set the speed field in the *cpuinfo* section of the system page.
- *cycles_freq* — the frequency at which the value returned by *ClockCycles()* increments. It's also used to set the *cycles_per_sec* field in the *qtime* section of the system page.
- *timer_freq* — the frequency at which the timer chip input runs. It's also used to set the *timer_rate* and *timer_scale* values of the *qtime* section of the system page.

If a variable is zero when it comes time to set the field(s) on the system page, the library code attempts to deduce the proper value by using one of the other frequency variables. Which one it uses depends on the particular CPU and hardware.

-I *flag*

Enable kernel restoration as part of IFS restoration. The *flag* is 0 to disable checksum verification, or 1 to enable it.

If checksum verification is enabled and fails, the entire image is reloaded.

Even if the IFS checksum verification is disabled, a checksum is still performed on the IFS Restoration internal data structure (approximately 32 bytes) to ensure at least some data integrity.

For more information, see *Reloadable Image Filesystems* in the QNX Neutrino technotes.

-i *ifs2_size*[,*flags*][,*paddr_src*][,*paddr_dst*]

Enable secondary IFS restoration.

The arguments are:

ifs2_size

The size of the secondary IFS (note: this can be larger than the actual size).

flags

Copyright © 2014, QNX Software Systems Limited

- Not specified — load the secondary IFS but don't try to restore on wake-up
- R — load the secondary IFS and restore
- K or RK — load the secondary IFS and restore with a checksum

paddr_src

- Not specified — the secondary IFS is located in flash after the primary IFS
- Specified — the secondary IFS is located at the physical address specified

paddr_dst

- Not specified — the secondary IFS will be copied to a default location in RAM
- Specified — the secondary IFS will be copied to the physical address specified (choose an address in a "safe" place, such as at the end of RAM away from where the primary image is copied)

If the checksum is enabled and fails, the entire secondary IFS is reloaded.

Even if the secondary IFS checksum is disabled, a checksum is still performed on the IFS Restoration internal data structure (approximately 16 bytes) to ensure at least some data integrity.

For more information, see *Reloadable Image Filesystems* in the QNX Neutrino technotes.

-j *addr*

For use with JTAG/hardware debuggers.

Reserve 4 bytes of RAM at the *physical* address specified by *addr*, and copy the *physical* address of the location of the system page to *addr* in RAM so that it can be retrieved by a hardware debugger.

-K *channel*[*.channel_opts*]

Specify an output channel for kernel debugger information. The format of this option and the default value vary from board to board.

-N *hostname*

Copyright © 2014, QNX Software Systems Limited

Specify the node name. The default is the local host.

-o *hundred_loop,overhead*

Specify the calibration data (100 loop time and overhead) to store in the system page. In order to make startup faster and reduce jitter, *nanospin_calibrate()* tries to read the calibration data from values stored in the system page.

-P *max_CPUs*

Specify the maximum number of processors to activate in a multicore system. This is useful for testing how well your application runs on a system with fewer CPUs. This option requires *procnto-smp* (p. 1586) instead of procnto to have an effect.

-R *size*[,*align*]

Remove *size* memory from system use, optionally specifying the alignment. This is useful for testing in a restricted-memory environment. The size and and alignment are in bytes, unless followed by one of K (kilobytes), M (megabytes), or G (gigabytes).

-r *addr,size*[,*flag*]

Remove *size* memory from system use starting at *addr*.

The *flag* is an optional argument used to specify if the memory should be cleared:

Flag:	Memory:
None	Clears to 0
0	Clears to 0
1	Does not clear

-S [~]*section*

Turn on (or, if you use ~*section*, off) output of the specified *syspage* section's information. Use this to restrict the amount of *syspage* information. For more information, see the description of *print_syspage()* in the Customizing Image Startup Programs chapter of *Building Embedded Systems*.

-T

Prevent the startup program from setting the SYSPAGE_ENTRY(qtime)->boot_time field. If this field is 0 the first time you call *ClockTime()* to

Copyright © 2014, QNX Software Systems Limited

change the time of day, the kernel sets it to the appropriate value. This is useful if the RTC hardware isn't in UTC.

-v[v]...

Be verbose. More `v` characters cause even more verbosity.

-Z

Enable tickless operation. In this mode, when the system is idle, the clock tick is "turned off" until just after the next active timer is to fire. For more information, see "Clocks, timers, and power management" in the Tick, Tock: Understanding the Microkernel's Concept of Time chapter of the QNX Neutrino *Programmer's Guide*.

Options for ARM startups

The following options are supported in the startup programs for ARM targets:

-w *value*

Set the cache policy:

- -wb sets write-back, read-allocate (no write-allocate) caching
- -wa sets write-back, write-allocate caching
- -wt sets write-through caching

The actual behavior depends on the processor; not all processors implement all these types of caching. If the processor doesn't implement the requested option, the default cache policy is used.

The supported cache policies are specified in the processor configuration file, `armv_chip_`*xxxx*`.c`:

This field:	Specifies PTE encodings for the:
pte	Default cache policy
pte_wb	Write-back (-wb) policy
pte_wa	Write-allocate (-wa) policy
pte_wt	Write-through (-wt) policy

The configuration must specify at least the default (*pte*) field. Any unsupported policy should specify 0 in the appropriate field, and if that policy is requested via the -w option, it's ignored, and the default policy is used.

Options for x86 startups

The following options are supported in the startup programs for x86 targets:

Copyright © 2014, QNX Software Systems Limited

-B

By default, x86 startups use the Advanced Control and Power Interface (ACPI) table to determine the number of logical CPUs on hyperthreaded systems. Use this option to avoid checking for ACPI in the case of buggy BIOSs; if ACPI isn't present or you specify -B, the startup uses the Intel Multiprocessor Specification to determine the number of CPUs.

-x

Enable extended addressing. This lets you access physical addresses above 4 GB.

This option has an effect only if the CPU supports more than 32 address lines. On x86 CPUs, extended addressing is supported if the X86_CPU_PAE bit is on in the SYSPAGE_ENTRY(cpuinfo)->flags. For more information, see "Structure of the system page" in the Customizing Image Startup Programs chapter of *Building Embedded Systems*.

Copyright © 2014, QNX Software Systems Limited

startup-apic, startup-apic-32

Startup for Intel Advanced Programmable Interrupt Controller (APIC) systems (QNX Neutrino)

Syntax:

```
startup-apic [-ABb] [-D channel[.channel_opts]]
    [-F [~]value]
    [-f [cpu_freq][,[cycles_freq][,timer_freq]]]
    [-I flag] [-i ifs2_size[,flags][,paddr_src][,paddr_dst]]
    [-j addr] [-K channel[.channel_opts]]
    [-N hostname] [-P max_cpus]
    [-R size[,align]] [-r addr,size[,flag]]
    [-S [~]section] [-s size] [-T] [-v[v]...] [-x] [-z]

startup-apic-32 [-ABb] [-D channel[.channel_opts]]
    [-F [~]value]
    [-f [cpu_freq][,[cycles_freq][,timer_freq]]]
    [-I flag] [-i ifs2_size[,flags][,paddr_src][,paddr_dst]]
    [-j addr] [-K channel[.channel_opts]]
    [-N hostname] [-P max_cpus]
    [-R size[,align]] [-r addr,size[,flag]]
    [-S [~]section] [-s size] [-T] [-v[v]...] [-x] [-z]
```

Runs on:

QNX Neutrino

Targets:

x86

Options:

In addition to the *generic* `startup-*` (p. 1848) and *x86-specific* (p. 1852) options, `startup-apic` and `startup-apic-32` support the following options:

-b

Don't reserve the bottom 4 KB of memory for virtual 8086 mode. This provides an extra 4 KB of memory for system use.

-s *size*

Copy the given amount of video card ROM into RAM, and set the x86 page tables to refer to the RAM copy rather than the ROM. The size is in bytes,

Copyright © 2014, QNX Software Systems Limited

unless followed by one of K (kilobytes), M (megabytes), or G (gigabytes). Specifying the -s option causes the following call:

```
x86_pcbios_shadow_rom( 0xc0000, size );
```

For more information, see "The startup library" in the Customizing Image Startup Programs chapter of *Building Embedded Systems*.

-z

Use the 8254 as system clock instead of the High Precision Event Timer (HPET), which is the default on Intel-based platforms. For more information, see below.

Debug channels

The debug channel specified with the -D and -K options can be:

8250[.*port*[^*shift*][.*baud*[.*clock*[.*divisor*]]]]

Use a generic 8250-compatible serial chip, with:

port

Specify the I/O port base address for the 8250, in hexadecimal. The default is 3f8.

shift

Specify the spacing between the I/O registers, in 2^{shift} bytes. The default is 0.

baud

Specify the baud rate for the debug channel. The default is 57600.

clock

Specify the clock rate (in Hz) input to the chip. The default is 1843200.

divisor

Specify the divisor used on the clock rate by the chip. The default is 16.

console

Use the PC console.

Copyright © 2014, QNX Software Systems Limited

You can skip options by leaving out the data associated with that part. For example, if you want to send the debugging output to an 8250 chip using 9600 baud, use:

```
-D 8250..9600
```

The default -D and -K settings are:

```
-D console
-K 8250.3f8^0.57600.1843200.16
```

Description:

The `startup-apic` and `startup-apic-32` programs are the startup for boards that support Intel Advanced Programmable Interrupt Controllers (APIC). They support Message Signaled Interrupts (MSI) and Extended MSI (MSI-X). They're similar, except that `startup-apic` uses 64-bit physical addresses, and `startup-apic-32` uses 32-bit ones.

If you're running `startup-apic` or `startup-apic-32`, you must use *`pci-bios-v2`* (p. 1455) instead of `pci-bios`, but it must still be called `pci-bios` in order for the enumerators to work correctly. In your buildfile, add `pci-bios-v2` like this:

```
pci-bios=pci-bios-v2
```

To assign MSI or MSI-X interrupts in a driver, use the `PCI_USE_MSI` or `PCI_USE_MSIX` flag when you call *pci_attach_device()*. For more information, see its entry in the QNX Neutrino *C Library Reference*.

By default on Intel-based platforms, `startup-apic` and `startup-apic-32` use High Precision Event Timer 0 (instead of the 8254) for the system clock. You can tell they're doing so in a couple of ways:

- With `startup-apic -v` or `startup-apic-32 -v`, you will see:

```
MSI interrupt = 0x00000100
MSI vector no = 78 0x0000004e
MSI vec count = 177
HPET0 selected for system clock on IRQ 2
Loading IFS...decompressing...done
lapic_configure(cpu=1)

System page at phys:00011000 user:fed14000 kern:fed16000
```

- With `pidin`, you will see:

```
# pidin syspage=qtime
Header size=0x0000009c, Total Size=0x000008e0, #Cpu=2, Type=0
Section:qtime offset:0x00000160 size:0x00000060
  boot:495d89cb CPS:000000005a82b720 rate/scale:69841279/-15
intr:2
```


Copyright © 2014, QNX Software Systems Limited

The 69841279 femptosecond clock period indicates that the HPET is being used. With the 8254 clock, the value is approximately 823 x 10^6 femptoseconds (one order of magnitude less resolution).

In order to prevent the sharing of the HPET0 interrupt with PCI devices, it's configured in legacy interrupt mode and so uses IRQ 2. This configuration also configures HPET1 in legacy interrupt mode on IRQ8, which eliminates the use of the RTC interrupts. The QNX Neutrino RTOS doesn't use RTC interrupts; if you need to use them, use the -z option to `startup-apic` or `startup-apic-32` to revert to the 8254 as system clock. Other HPET timers (if the system supports them) aren't affected by the legacy interrupt routing of HPET0 and HPET1.

Examples:

Direct debug output to the console:

```
startup-apic -Nnode120 -vvvv -Dconsole
```

Direct debug output to the first serial port (making sure the baud rate was set to 115200 on the receiving side):

```
startup-apic -Nnode120 -vvvv -D8250..115200
```

Direct debug output to the serial port at `2f8`:

```
startup-apic -Nnode120 -vvvv -D8250.2f8.115200
```

Copyright © 2014, QNX Software Systems Limited

startup-bios, startup-bios-32

Startup for PC-compatible systems with a BIOS (QNX Neutrino)

Syntax:

```
startup-bios [-ABb] [-D channel[.channel_opts]]
    [-F [~]value]
    [-f [cpu_freq][,[cycles_freq][,timer_freq]]]
    [-I irq] [-j addr] [-K channel[.channel_opts]] [-L]
    [-N hostname] [-P max_cpus]
    [-R size[,align]] [-r addr,size[,flag]]
    [-S [~]section] [-s size] [-T] [-v[v]...] [-x]

startup-bios-32 [-ABb] [-D channel[.channel_opts]]
    [-F [~]value]
    [-f [cpu_freq][,[cycles_freq][,timer_freq]]]
    [-I irq] [-j addr] [-K channel[.channel_opts]] [-L]
    [-N hostname] [-P max_cpus]
    [-R size[,align]] [-r addr,size[,flag]]
    [-S [~]section] [-s size] [-T] [-v[v]...] [-x]
```

Runs on:

QNX Neutrino

Targets:

x86 with a PC-compatible BIOS

Options:

In addition to the *generic* `startup-*` (p. 1848) and *x86-specific* (p. 1852) options, `startup-bios` and `startup-bios-32` support the following options:

-b

Don't reserve the bottom 4 KB of memory for virtual 8086 mode. This provides an extra 4 KB of memory for system use.

-I ***irq***

Make *irq* the highest-priority hardware interrupt in the system. You can specify a number from 0 through 7 (the default is 3).

This overrides the default -I option.

-L

Copyright © 2014, QNX Software Systems Limited

Enable support for the Local APIC interrupts (X86_INTR_APIC_* definitions in <x86/intr.h>).

-s *size*

Copy the given amount of video card ROM into RAM, and set the x86 page tables to refer to the RAM copy rather than the ROM. The size is in bytes, unless followed by one of K (kilobytes), M (megabytes), or G (gigabytes). Specifying the -s option causes the following call:

```
x86_pcbios_shadow_rom( 0xc0000, size );
```

For more information, see "The startup library" in the Customizing Image Startup Programs chapter of *Building Embedded Systems*.

Debug channels

The debug channel specified with the -D and -K options can be:

8250[**.***port*[**^***shift*][**.***baud*[**.***clock*[**.***divisor*]]]]

Use a generic 8250-compatible serial chip, with:

port

Specify the I/O port base address for the 8250, in hexadecimal. The default is 3f8.

shift

Specify the spacing between the I/O registers, in 2^{shift} bytes. The default is 0.

baud

Specify the baud rate for the debug channel. The default is 57600.

clock

Specify the clock rate (in Hz) input to the chip. The default is 1843200.

divisor

Specify the divisor used on the clock rate by the chip. The default is 16.

console

Use the PC console.

Copyright © 2014, QNX Software Systems Limited

You can skip options by leaving out the data associated with that part. For example, if you want to send the debugging output to an 8250 chip using 9600 baud, use:

```
-D 8250..9600
```

The default -D and -K settings are:

```
-D console
-K 8250.3f8^0.57600.1843200.16
```

Description:

The `startup-bios` and `startup-bios-32` programs are responsible for probing PC hardware resources using the BIOS. They're similar, except that `startup-bios` uses 64-bit physical addresses, and `startup-bios-32` uses 32-bit ones.

If you want to use Message Signaled Interrupts (MSI) or Extended MSI (MSI-X), use *`startup-apic`* (p. 1854) instead of `startup-bios` in your build file. You must use `startup-apic` in conjunction with *`pci-bios-v2`* (p. 1455).

Examples:

Direct debug output to the console:

```
startup-bios -Nnode120 -vvvv -Dconsole
```

Direct debug output to the first serial port (making sure the baud rate was set to 115200 on the receiving side):

```
startup-bios -Nnode120 -vvvv -D8250..115200
```

Direct debug output to the serial port at `2f8`:

```
startup-bios -Nnode120 -vvvv -D8250.2f8.115200
```

For more examples, see `${`***`QNX_TARGET`***`}/x86/build`.

Copyright © 2014, QNX Software Systems Limited

strings

Find printable strings in files (POSIX)

Syntax:

strings_variants [*options*] *file*...

where *strings_variant* depends on the target platform, as follows:

Target platform	*strings_variant*
ARMv7	`ntoarmv7-strings`
x86	`ntox86-strings`

Runs on:

Linux, Microsoft Windows

Description:

For each *file* given, `strings` prints the printable character sequences that are at least 4 characters long (or the number given with the options) and are followed by an unprintable character. By default, it prints only the strings from the initialized and loaded sections of object and ELF files; for other types of files, it prints the strings from the whole file.

The `strings` utility is mainly useful for determining the contents of nontext files.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

strip

Remove unnecessary information from executable files (POSIX)

Syntax:

strip_variant [*options*] *objfile*...

where *strip_variant* depends on the target platform, as follows:

Target platform	*strip_variant*
ARMv7	`ntoarmv7-strip`
x86	`ntox86-strip`

Runs on:

Linux, Microsoft Windows

Description:

The `strip` utility discards all symbols from object files *objfile*. The list of object files may include archives. At least one object file must be given.

This utility modifies the files named in its argument instead of writing modified copies under different names.

For detailed documentation, see the GNU website at *http://www.gnu.org/*.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

stty

Set tty attributes (POSIX)

Syntax:

```
stty [-a|-g] [operands] [< device]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Display all settings.

-g

Display in "getable" form.

Description:

The `stty` utility sets and/or reports terminal `I/O` characteristics for the device that is its standard input. If no operands are specified, `stty` displays the settings. If operands are given, then `stty` changes the terminal state to reflect those settings.

Terminal settings fall into two major categories:

edit mode

The user can edit the input data. It's made available to programs only when a `CR` is pressed. Output data is normally presented in a human-readable format.

raw mode

All data flows to and from the terminal with little extra processing.

If you're at a shell prompt, your terminal is probably in the default system edit mode. A full-screen program such as an editor, on the other hand, typically puts the terminal into raw mode.

Normally, `stty` displays only significant settings relative to the system default settings for edit or raw, and displays only the defined control characters. If -a or -g is specified, then `stty` displays *all* the settings.

Copyright © 2014, QNX Software Systems Limited

The `stty` utility manages a very large number of potential terminal attributes and control characters. Most of these parameters are very device-specific and seldom need to be changed by the user. Programs often change these terminal attributes in the course of their operation, and upon occasion (such as abnormal termination) may leave the terminal settings in an unknown state. The `stty` +edit option is a convenient method of restoring a terminal to a usable state.

Supported operands

The tables below list the operands supported by `stty`. In these tables, the following conventions are used:

number

A decimal integer number (e.g. 9600).

name

A string of characters (e.g. `vt100`).

value

A single character (e.g. ~) *or* a 2-digit hexadecimal number (e.g. `1B`) *or* one of the following character pairs:

Char pair	Hex
^-	00 (undefined)
^A *to* ^Z	01 to 1A
^[	1B
^\	1C
^]	1D
^^	1E
^_	1F
^?	7F

Some options can start with either + or -:

+

Turn the option on.

-

Turn the option off.

Copyright © 2014, QNX Software Systems Limited

If you don't specify + or -, + is assumed.

Also note that the = character is optional in operands of the form *keyword=value*.

The following descriptions give the action taken when the option is *on*.

Line control parameters

Parameter	Defines
baud=*number*	Input and output baud rates
ispeed=*number*	Input baud rate
ospeed=*number*	Output baud rate
par=none	No parity (same as -parenb)
par=odd	Odd parity (same as +parenb, +parodd, -parstk)
par=even	Even parity (same as +parenb, -parodd, -parstk)
par=mark	Mark parity (same as +parenb,+parodd, +parstk)
par=space	Space parity (same as +parenb, -parodd, +parstk)
bits=5	5-bit characters
bits=6	6-bit characters
bits=7	7-bit characters
bits=8	8-bit characters
stopb=2	2-stop bits
stopb=1	1-stop bits
{+|-}parenb	Enable parity
+parodd	Odd parity
-parodd	Even parity
{+|-}parstk	Stick parity
+cs5	Same as bits=5
+cs6	Same as bits=6
+cs7	Same as bits=7

Copyright © 2014, QNX Software Systems Limited

Parameter	Defines
+cs8	Same as bits=8
+cstopb	Same as stopb=2
-cstopb	Same as stopb=1
number	Same as baud=*number*
+evenp	Same as par=even, bits=7
-evenp	Same as par=none, bits=8
+parity	Same as par=even, bits=7
-parity	Same as par=none, bits=8
+oddp	Same as par=odd, bits=7
-oddp	Same as par=none, bits=8
{+|-}hupcl	Hangup on last close
{+|-}hup	Same as hupcl
{+|-}cread	Enable receiver
{+|-}clocal	Assume no modem control
{+|-}ihflow	Enable hardware input flow control
{+|-}ohflow	Enable hardware output flow control
{+|-}isflow	Enable software input flow control
{+|-}osflow	Enable software output flow control
{+|-}ihpaged	Input is paged by hardware flow control
{+|-}ohpaged	Output is paged by hardware flow control
{+|-}ispaged	Input is paged by software flow control
{+|-}ospaged	Output is paged by software flow control
rows=*value*[,*value*]	The number of rows and (optionally) columns of the terminal

Input processing parameters

Parameter	Defines
{+|-}ignbrk	Ignore received hardware breaks
{+|-}brkint	Generate `SIGINT` upon break

Copyright © 2014, QNX Software Systems Limited

Parameter	Defines
{+|-}ignpar	Ignore parity errors
[-]imaxbel	Beep and don't flush a full input buffer on a character
{+|-}parmrk	Mark parity errors
{+|-}inpck	Enable software parity checking
{+|-}istrip	Strip 7th bit from received characters
{+|-}inlcr	Map NL into CR on input
{+|-}onlcr	Map NL into CR on output
{+|-}igncr	Ignore received CR
{+|-}icrnl	Map CR into NL on input
{+|-}ixon	Same as osflow
{+|-}ixoff	Same as isflow
{+|-}isig	Generate signals upon receipt of special characters
{+|-}icanon	Enable input line editing
{+|-}iexten	Enable “extra” special characters
{+|-}echo	Echo received characters
{+|-}echoctl	Echo control characters in hat notation (e.g ^c)
{+|-}echoe	Erase character erases displayed character
{+|-}echok	Echo a newline after a kill character
{+|-}echoke	Kill character erases displayed line
{+|-}echonl	Echo NL, even if ECHO is off
{+|-}noflsh	Don't flush I/O after INTR, QUIT, or SUSP
min=*number*	Minimum characters required to satisfy raw input
time=*number*	Timeout value for raw input
{+|-}tostop	Send SIGTTOU for background output.
+nl	Same as +icrnl

Copyright © 2014, QNX Software Systems Limited

Parameter	Defines
-nl	Same as -icrnl, -inlcr, -igncr
+sane	Reset all parameters to sane values based on current mode (*edit*/*raw*)
+fix	Same as +sane
+edit	Reset parameters to system default *edit* mode
+flush	Flush all pending input and output
+raw	Reset parameters to system default *raw* mode

Output processing parameters

Parameter	Defines
{+|-}opost	Post-process output data

Special control characters

Parameter	Defines
discard=*value*	Discard character
dsusp=*value*	Character for sending a terminal stop signal once input is flushed
eof=*value*	End-of-file character
eol=*value*	End-of-line character
eol2=*value*	Alternative end-of-line character
erase=*value*	Delete previous character
fwd=*value*	Forwarding or framing character; if this character is set, then *readcond()* returns when either the read length is reached or the FWD character is found in the data stream. For more information, see the entry for *readcond()* in the QNX Neutrino *C Library Reference*.
intr=*value*	Generate `SIGINT` character
kill=*value*	Delete entire line character

Copyright © 2014, QNX Software Systems Limited

Parameter	Defines
lnext=*value*	Character for entering the next character quoted
quit=*value*	Generate SIGQUIT character
reprint=*value*	Character for redrawing the current line
susp=*value*	Generate SIGTSTP character
swtch=*value*	Character for switching to a different shell layer
stop=*value*	Stop output
start=*value*	Resume output
+ek	Resets ERASE and KILL to system defaults
werase=*value*	Character for erasing the last word typed

Extended line-editing character sequences

The QNX Neutrino RTOS supports multicharacter sequences to support editing capabilities in addition to the basic “erase” and “kill” functions. All of these sequences are assumed to start with an (up to) 4-character *prefix*, a single-character *action*, followed by an (up to) 4-character *suffix*. Typically, the cursor keys on your terminal are used for these functions.

Parameter	Meaning
+load	Set editing keys based on currently defined terminal type
term=*name*	Set editing keys for the specified type of terminal
pr1=*value*	First character of *prefix*
pr2=*value*	Second character of *prefix*
pr3=*value*	Third character of *prefix*
pr4=*value*	Fourth character of *prefix*
sf1=*value*	First character of *suffix*
sf2=*value*	Second character of *suffix*
sf3=*value*	Third character of *suffix*
sf4=*value*	Fourth character of *suffix*

Copyright © 2014, QNX Software Systems Limited

Action characters

The following are *action* characters when preceded by *prefix* (subsequent *suffix* are discarded).

Parameter	Meaning
up=*value*	Recall previous line
down=*value*	Recall next line
left=*value*	Move cursor left
right=*value*	Move cursor right
ins=*value*	Toggle insert mode
del=*value*	Delete current character
rub=*value*	Delete previous character
can=*value*	Delete entire line character
home=*value*	Move cursor to beginning of line
end=*value*	Move cursor to end of line

Examples:

Display settings of the terminal to which `stty` is attached:

```
stty
```

Display settings of a specified device:

```
stty < /dev/ser1
```

Change the baud rate of a specified device:

```
stty baud=1200 < /dev/ser1
```

Put terminal into a fixed, sane state:

```
stty +sane
```

Set editing keys to `VT100` values:

```
stty term=vt100
```

Restore settings from a shell variable:

```
stty $saveterm
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

Success.

>0

An error occurred.

Caveats:

The `stty` utility quietly accepts all of the documented options, but some of the current managers might not support them.

Copyright © 2014, QNX Software Systems Limited

su

Switch user ID (UNIX)

Syntax:

```
su [-c] [[-] userid [arguments]]
```

Runs on:

QNX Neutrino

Options:

-c

Pass the specified list of arguments to the shell.

userid

Switch to the specified user ID. If preceeded by the - argument, *userid*'s login scripts will run, setting up the shell environment as if you had actually logged in as the specified user.

Description:

The `su` utility lets you temporarily become another user, then return to your regular *userid*.

The `su` utility requests the password of the given *userid* (`root`, by default) and changes to that *userid*, invoking its shell but modifying only essential elements of the environment. Only the ***HOME***, ***PATH***, and possibly ***SHELL*** environment variables are changed, but the new shell has the rights and privileges of the user specified. The new *userid* remains in effect until this shell exits.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Files:

/bin/sh

Korn shell command interpreter.

Copyright © 2014, QNX Software Systems Limited

/dev/null

The bit bucket.

/etc/.pwlock

This file is used to lock password files when modifications are taking place.

/etc/acclog

Logs system accounting information.

/etc/default/profile

The `passwd` utility copies this file as a user's initial `.profile` when it creates a new account.

/etc/group

Defines the known groups for the system.

/etc/passwd

Defines the valid user IDs on the system.

/etc/shadow

Contains encoded versions of the actual passwords for user accounts. The passwords themselves aren't stored in the `/etc/passwd` file.

/usr/adm/sulog

Records all `su` activity.

Copyright © 2014, QNX Software Systems Limited

sync

Update filesystems to match cached data (UNIX)

Syntax:

sync

Runs on:

QNX Neutrino

Options:

None.

Description:

The sync utility forces the filesystem manager to begin flushing all modified in-memory inodes and all previously unwritten system buffers to disk. This ensures that all file modifications up to that point are scheduled to be saved.

The sync utility usually returns before the operation is complete. After executing sync, you must allow sufficient time for the driver queues to drain before you may assume that the data is safely on disk. This delay depends on the speed of your disk(s), the number of buffers that must be drained, and how active your system is at the time. It is unusual for the operation to take more than four to five seconds with typical IDE drives and a large (2 megabyte) cache, if a large portion of the cache needs to be flushed. With faster disks (e.g. SCSI), it usually completes in under two seconds. If significant amounts of data are being flushed to a floppy disk, it could take many tens of seconds.

Examples:

Synchronize the local filesystem:

```
sync
```


Copyright © 2014, QNX Software Systems Limited

sysctl

Get or set the state of the socket manager

Syntax:

```
sysctl [-dne] [-x[x]|-r] variable ...
sysctl [-ne] [-q] -w variable=value ...
sysctl [-dne] -a
sysctl [-dne] -A
sysctl [-ne] -M
sysctl [-dne] [-q] -f file
```

Runs on:

QNX Neutrino

Options:

-A

List all the known MIB names, including tables. Those with string or integer values are displayed as they would be with the -a option; for the table values, the name of the utility to retrieve them is given.

-a

List all the currently available string or integer values.

-d

Display descriptions of the selected nodes. The default is to display their values.

-e

Separate the name and value of the variables with an equals sign (=). This format is useful when you're producing output to be given as input to `sysctl`. The default is to use an equals sign with a space on either side. This option is ignored if you also specify the -n option, or if you're setting a variable.

-f *file*

Read and process the specified file. The format of the file is as follows:

- Blank lines and comments (beginning with #) are ignored.
- You can use a backslash to escape the end of the line.

Copyright © 2014, QNX Software Systems Limited

- Remaining lines are processed similarly to command-line arguments of the form *name* or *name=value*.

This option implies the -w option. Any name arguments on the command line are ignored.

-M

Display the MIB instead of any of the actual values contained in the MIB. This causes the entire MIB to be displayed unless you also give specific MIB arguments or the -f *file* option.

-n

Don't display the field name; display only its value. You'll find this option useful when you're setting shell variables. For example, to save the IP TTL value in the variable *ipttl*, type the following:

```
set ipttl=`sysctl -n net.inet.ip.ttl`
```

-q

Be quiet; display nothing when setting variables, unless an error occurs.

-r

Display values in their raw binary forms as retrieved directly. You can use this option to retrieve some additional nodes that `sysctl` can't display directly. This option conflicts with the -x option.

-w *name=value*

Set the value for the given MIB name.

-x

Display the requested value in a hexadecimal representation instead of its regular form. If you specify this option more than once, the output for each value includes the hexadecimal offset, two sets of eight columns of hexadecimal bytes, then a vertical bar (`|`), followed by the ASCII representation of the bytes. This option conflicts with the -r option.

Description:

The `sysctl` utility retrieves the state of the socket manager and allows processes with appropriate privilege to set the state. The variable to be retrieved or set is described using a *Management Information Base* (MIB) style name, described as a dotted set of components.

Copyright © 2014, QNX Software Systems Limited

The information available from `sysctl` consists of integers, strings, and tables. You can retrieve tabular information only by using special-purpose programs such as *arp* (p. 46) and *netstat* (p. 1339).

The variables that are available to you depend on what you're running on your machine; the table below shows the variables that are likely of most interest. For information about determining the meaning of other variables, see *sysctl()* in the QNX Neutrino *C Library Reference*.

A process with appropriate privilege can change the value of all these variables except those marked as read-only. All values are integers unless otherwise indicated.

`kern.clockrate` (read only)

A `struct clockinfo` that contains the clock, statistics clock and profiling clock frequencies, the number of microseconds per Hz tick, and the clock skew rate.

`kern.mbuf.mblowat`

The `mbuf` low water mark.

`kern.mbuf.mclbytes`

The `mbuf` cluster size.

`kern.mbuf.mcllowat`

The `mbuf` cluster low water mark.

`kern.mbuf.msize` (read only)

The `mbuf` base size.

`kern.mbuf.nmbclusters`

The limit on the number of `mbuf` clusters. You can only increase this limit, and only on machines with direct-mapped pool pages.

`kern.sbmax`

The maximum socket buffer size.

`net.inet.arp.down`

The failed ARP entry lifetime.

`net.inet.arp.keep`

The valid ARP entry lifetime.

`net.inet.arp.prune`

The ARP cache pruning interval.

Copyright © 2014, QNX Software Systems Limited

net.inet.arp.refresh

The ARP entry refresh interval.

net.inet.ip.allowsrcrt

Allow (1) or drop (0) all source-routed packets.

net.inet.ip.directed-broadcast

Enable (1) or disable (0) directed-broadcast.

net.inet.ip.do_loopback_cksum

Compute (1) or don't compute (0) checksums on loopback.

net.inet.ip.forwarding

Disable (0) or enable (1) IP forwarding. If this is enabled, the host acts as a router.

net.inet.ip.forwsrcrt

Forward source-routed packets.

net.inet.ip.maxflows

The maximum number of IP flows allowed.

net.inet.ip.mtudisc

Allow (1) or disallow (0) path MTU discovery.

net.inet.ip.redirect

Allow (1) or disallow (0) send ICMP redirections when forwarding. This option is ignored unless the host is routing IP packets. Normally, this option should be enabled on all systems.

net.inet.ip.subnetsarelocal

Treat (1) or don't treat (0) subnets as local addresses.

net.inet.ip.ttl

The maximum time-to-live (hop count) value for an IP packet sourced by the system. This value applies to normal transport protocols, not to ICMP.

net.inet.tcp.congctl.available

A string that lists the available TCP congestion-control algorithms.

net.inet.tcp.congctl.selected

Copyright © 2014, QNX Software Systems Limited

A string that contains the name of the currently selected TCP congestion-control algorithm.

net.inet.tcp.do_loopback_cksum

Compute (1) or don't compute (0) checksums on loopback.

net.inet.tcp.keepcnt

The keepalive count.

net.inet.tcp.keepidle

The keepalive idle time, in milliseconds.

net.inet.tcp.keepintvl

The keepalive probe interval, in milliseconds.

net.inet.tcp.mssdflt

The default maximum segment size.

net.inet.tcp.recvspace

The default size of the receive buffer.

net.inet.tcp.sack.enable

Enable (1) or disable (0) *RFC 2018* Selective ACKnowledgements.

net.inet.tcp.sack.globalholes (read only)

The global number of TCP SACK holes.

net.inet.tcp.sack.globalmaxholes

The global maximum number of TCP SACK holes.

net.inet.tcp.sack.maxholes

The maximum number of TCP SACK holes allowed per connection.

net.inet.tcp.sendspace

The default size of the send buffer.

net.inet.tcp.slowhz (read only)

The units for `tcp.keepidle` and `tcp.keepintvl`; those variables are in ticks of a clock that ticks `tcp.slowhz` times per second. (That is, you must divide their values by the value of `tcp.slowhz` to get times in seconds.)

net.inet.tcp.win_scale

RFC 1323 window scaling.

Copyright © 2014, QNX Software Systems Limited

net.inet.udp.do_loopback_cksum

Compute (1) or don't compute (0) checksums on loopback.

net.inet.udp.recvspace

The default size of the receive buffer.

net.inet.udp.sendspace

The default size of the send buffer.

net.inet6.ip6.forwarding

Disable (0) or enable (1) IP forwarding. If this is enabled, the host acts as a router.

net.inet6.ip6.redirect

Allow (1) or disallow (0) send ICMP redirections when forwarding. This option is ignored unless the host is routing IP packets. Normally, this option should be enabled on all systems.

net.inet6.tcp6.do_loopback_cksum

Compute (1) or don't compute (0) checksums on loopback.

net.inet6.tcp6.keepcnt

The keepalive count.

net.inet6.tcp6.keepidle

The keepalive idle time, in milliseconds.

net.inet6.tcp6.keepintvl

The keepalive probe interval, in milliseconds.

net.inet6.tcp6.recvspace

The default size of the receive buffer.

net.inet6.tcp6.sack.enable

Enable (1) or disable (0) *RFC 2018* Selective ACKnowledgements.

net.inet6.tcp6.sack.globalholes (read only)

The global number of TCP SACK holes.

net.inet6.tcp6.sack.globalmaxholes

The global maximum number of TCP SACK holes.

net.inet6.tcp6.sack.maxholes

Copyright © 2014, QNX Software Systems Limited

The maximum number of TCP SACK holes allowed per connection.

net.inet6.tcp6.sendspace

The default size of the send buffer.

net.inet6.tcp6.slowhz (read only)

The units for `tcp.keepidle` and `tcp.keepintvl`; those variables are in ticks of a clock that ticks `tcp.slowhz` times per second. (That is, you must divide their values by the value of `tcp.slowhz` to get times in seconds.)

net.inet6.udp6.do_loopback_cksum

Compute (1) or don't compute (0) checksums on loopback.

net.inet6.udp6.recvspace

The default size of the receive buffer.

net.inet6.udp6.sendspace

The default size of the send buffer.

You can set variables permanently by setting them in a file such as `/etc/sysctl.conf`, and then starting `sysctl` using that file. For example, you could add this command:

```
sysctl -f /etc/sysctl.conf
```

to your system's `/etc/rc.d/rc.local` file.

Examples:

Check to see if the UDP checksum is enabled:

```
sysctl net.inet.udp.checksum
```

Disabling UDP checksums is strongly discouraged.

Enable IP forwarding so that the host acts as a router:

```
sysctl -w net.inet.ip.forwarding=1
```

Copyright © 2014, QNX Software Systems Limited

/etc/syslog.conf

Configuration file for `syslogd`

Name:

`/etc/syslog.conf`

Description:

The `/etc/syslog.conf` file is the configuration file for the *`syslogd`* (p. 1885) daemon. It consists of lines with two fields:

Selector field

The types of messages and priorities to which the line applies.

Action field

The action to take if a message received by `syslogd` matches the selection criteria.

Use one or more tab characters to separate the selector and action fields.

The selectors are encoded as a *facility*, a dot (`.`), and a *level*, with no intervening whitespace. Both the *facility* and the *level* are case-insensitive.

The *facility* describes the part of the system generating the message, and is one of the following keywords:

- `auth`
- `authpriv`
- `cron`
- `daemon`
- `kern`
- `lpr`
- `mail`
- `mark`
- `news`
- `syslog`
- `user`
- `uucp`
- `local0` through `local7`.

Copyright © 2014, QNX Software Systems Limited

These keywords (with the exception of `mark`) correspond to the similar "LOG_" values specified to the *openlog()* and *syslog()* routines.

The *level* describes the severity of the message, and is a keyword from the following ordered (higher to lower) list:

- `emerg`
- `alert`
- `crit`
- `err`
- `warning`
- `notice`
- `info`
- `debug`

These keywords also correspond to the similar "LOG_" values specified to the *syslog()* routine.

For further descriptions of both the *facility* and *level* keywords and their significance, see *syslog()* in the QNX Neutrino *C Library Reference*.

If a received message matches the specified *facility* and is of the specified (or higher) *level*, then the action specified in the *action* field is taken.

You can specify multiple selectors for a single *action* by separating them with semicolon (`;`) characters.

Note that each selector can modify the ones preceding it.

You can specify multiple facilities for a single *level* by separating them with comma (`,`) characters.

You can use an asterisk (`*`) to specify all facilities or all levels.

The special *facility* `mark` receives a message at priority `info` every 20 minutes (see *syslogd* (p. 1885)).

The special *level* `none` disables a particular *facility*.

The *action* field of each line specifies the action to be taken when the *selector* field selects a message. The *action* field can take these forms:

- a pathname (beginning with a leading slash) — the selected messages are appended to the file
- a hostname (preceded by @) — the selected messages are forwarded to the `syslogd` program on the named host

Blank lines and lines whose first nonblank character is a hash (#) character are ignored.

Copyright © 2014, QNX Software Systems Limited

Examples:

A configuration file might appear as follows:

```
# Log all kernel messages, authentication messages of
# level notice or higher and anything of level err or
# higher to the console.
# Don't log private authentication messages!
*.err;kern.*;auth.notice;authpriv.none /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none        /var/log/messages

# The authpriv file has restricted access.
authpriv.*                            /var/log/secure

# Log all the mail messages in one place.
mail.*                                /var/log/maillog

# Everybody gets emergency messages, plus log them on
# another machine.
*.emerg                   *
*.emerg                   @arpa.berkeley.edu

# Root and Eric get alert and higher messages.
*.alert                   root,eric

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit            /var/log/spoolerr
```

Caveats:

The effects of multiple selectors aren't always intuitive. For example, `mail.crit,*.err` selects `mail` facility messages at the level of `err` or higher, *not* at the level of `crit` or higher.

Logging messages to users isn't currently implemented.

Copyright © 2014, QNX Software Systems Limited

syslogd

Log system messages

You must be `root` to run this daemon.

Syntax:

```
syslogd [-f config_file] [-m mark_interval]
        [-t threads]
```

Runs on:

QNX Neutrino

Options:

-f *config_file*

Specify the pathname of an alternate configuration file (the default is `/etc/syslog.conf`).

-m *mark_interval*

Select the number of minutes between “mark” messages (the default is 20 minutes).

-t *threads*

Set the maximum number of threads that `syslogd` should use (the default is 15).

Description:

The `syslogd` daemon reads and logs messages to the system console, log files, and other machines as specified by its configuration file.

The daemon reads its configuration file when it starts up and whenever it receives a hangup signal. For information on the format of the configuration file, see *`/etc/syslog.conf`* (p. 1882).

The messages sent to `syslogd` should consist of a single line, which may start with a facility/priority (as defined in `<syslog.h>`) in angle brackets (e.g. `"<5> hello"`). If the message doesn't specify a priority, it defaults to `LOG_USER|LOG_NOTICE` (`"<13>"`).

Copyright © 2014, QNX Software Systems Limited

The *syslog()* API (and the *logger* (p. 1119) utility, which uses *syslog()*) sends messages to `syslogd` by opening and writing to `/dev/log`.

If a log message is submitted to `/dev/log`, the entry includes the string `nto` instead of the local host name. If the log message is submitted via the socket interface, the entry includes the host name. If the message is redirected from another `syslogd` on another host, the entry includes the host name.

Files:

The `syslogd` daemon requires the following files:

`/etc/syslog.conf`

This configuration file contains the selection criteria and the action to be taken if a message received by syslogd matches the selection criteria.

`/etc/services`

This file specifies the Internet domain socket port that `syslogd` listens to.

The `syslogd` daemon also requires the `libsocket.so` shared library.

Environment variables:

`SYSLOG`

Used by clients to specify which node to look for `syslogd`.

Copyright © 2014, QNX Software Systems Limited

Chapter 21
T

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “T”.

Copyright © 2014, QNX Software Systems Limited

tail

Copy the last part of files (POSIX)

Syntax:

```
tail [-number] [-fl] [-c number | -n number] [file]...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-number

Deprecated; use -n *number* instead.

-c *number*

Copy the given number of bytes; see below.

-f

If the input file is a regular file (i.e. not a tty or FIFO), don't terminate after the last line of the input file has been copied, but enter a continuous loop. The `tail` utility then sleeps for approximately one second, then attempts to read and copy further bytes from the input file.

-l

("el") Measure the quantity of output in lines; this is the default unit of measure. Deprecated; use -n *number* instead.

-n *number*

Copy the given number of lines; see below.

file

The pathname of an input file. If you don't specify a file, the standard input is used.

Description:

The `tail` utility copies its input files to the standard output, beginning at the point in the files indicated by the -c or -n option. For both options, the *number* argument is a decimal integer whose sign specifies the location in the file to begin copying:

Copyright © 2014, QNX Software Systems Limited

If the sign is:	Then copying starts relative to the:
+	Beginning of the file
-	End of the file
Omitted	End of the file

If you don't specify a -c or -n option, the default is -n 10 (i.e. the last ten lines of the file).

If you use both -c and the deprecated -l option, the order of the options is important. If you specify:

```
tail -l -c 5 my_file
```

then `tail` copies 5 bytes. If you specify:

```
tail -c 5 -l my_file
```

it copies 5 lines.

When `tail` is applied to a nonseekable file (e.g. a tty), `tail` must maintain an internal buffer. This buffer is large enough to hold *at least* 10 lines of characters.

Examples:

You can use the -f option to monitor the growth of a file that is being written by a process. For example, the command:

```
tail -f fred
```

prints the last ten lines of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and terminated. As another example, the command:

```
tail -f -c 15 fred
```

prints the last 15 bytes of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and terminated.

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

tar

Read and write tape archive files (UNIX)

Syntax:

Create a new archive:

```
tar -c [-b blksize] [-f file] [-vw] [filename...]
```

Write named files to the end of an archive:

```
tar -r -f file [-b blksize] [-vw] [filename...]
```

List all the files contained in an archive:

```
tar -t [-f file] [-v] [filename]
```

Extract named files from an archive:

```
tar -x [-f file] [-lmovw] [filename...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-[0-7][lmh]

Specify the drive and density.

-A

Append tar files to an archive.

-B

Reblock as we read (for 4.2BSD pipes).

-b *blksize*

Specify the blocking factor for tape records. The default is 1; the maximum is 20. This option should be used only with raw magnetic tape archives. Normally, the block size is determined automatically when reading tapes.

-C=*dir*

Change to the directory *dir*.

-c

Copyright © 2014, QNX Software Systems Limited

Create a new archive. Writing begins at the beginning of the archive, instead of after the last file.

-d

Find the differences between the archive and the filesystem.

-F=*file*

Run the given script at the end of each tape (implies -M).

-f *file*

Specify the name of the archive to use instead of the default, which is standard output. If you specify the dash character (-) as a filename, `tar` writes to the standard output or reads from the standard input, whichever is appropriate for the options given. Thus, you can use `tar` as the head or tail of a pipeline.

-G

Handle the old GNU-format incremental backup.

-g

Handle the new GNU-format incremental backup.

-h

Dump instead the files that symlinks point to.

-i

Ignore zeroed blocks in archive (means EOF).

-K=*name*

Begin at file *name* in the archive.

-k

Don't overwrite existing files when extracting.

-L *num*

Change tape after writing *num* × 1024 bytes.

-l

(“el”) Report if all of the links to the files being archived cannot be resolved. If this option isn't specified, no error messages are written to the standard output. This option is valid only with the -c and -r options.

-M

Copyright © 2014, QNX Software Systems Limited

Create, list, or extract a multivolume archive.

-m

Don't restore modification times. The modification time of the file is the time of extraction. This option is invalid with the -t option.

-N=*date*

Store only the files that are newer than *date*.

-O

(“Oh”) Extract files to standard output.

-o

Write a V7 format archive.

-P

Don't strip leading slashes (/) from filenames.

-p

Extract all protection information.

-R

Show the block number within the archive with each message.

-r

Write named files to the end of the archive specified in the required -f *file* option.

-S

Handle sparse files efficiently.

-s

Sort the names to extract to match archive.

-T=*name*

Get the names to extract or create from the file, *name*.

-t

List the names of all of the files in the archive.

-U

Unlink each file prior to extracting over it.

Copyright © 2014, QNX Software Systems Limited

-u

Append only files that are newer than the copy in the archive.

-V=*name*

Create an archive with the volume name specified by *name*.

-v

Be verbose. Usually, `tar` works silently, but the -v option causes it to print the name of each file it processes, preceded by the option letter. With the -t option, -v gives more information about the archive entries than just the name.

-W

Attempt to verify the archive after writing it.

-w

Print the action to be taken, followed by the name of the file, then wait for the user's confirmation. If you enter a word beginning with `y`, the action is performed. Any other input means “no”. This option is invalid with the -t option.

-X=*file*

Exclude the globbing patterns listed in *file*.

-x

Extract named files from the archive. If a named file matches a directory whose contents had been written onto the archive, that directory is recursively extracted. If a named file in the archive doesn't exist on the system, the file is created with the same mode as the one in the archive, except that the set-user-id and set-group-id modes are set only if you have appropriate privileges.

If the files exist, their modes are not changed except as described above. The owner, group, and modification time are restored if possible. If no *filename* argument is given, the entire contents of the archive is extracted. Note that if several files with the same name are in the archive, the last one overwrites all earlier ones.

-z

Filter the archive through *gzip* (p. 921).

filename

Copyright © 2014, QNX Software Systems Limited

The pathname of the file to be archived.

Description:

The `tar` utility reads and writes archive files. For more information, see the GNU website at *http://www.gnu.org/*.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

The GNU `tar` is incompatible with the current POSIX standard and with `tar` programs that follow POSIX. For details, see the GNU documentation.

Examples:

Display a verbose listing of the archive members in `dist.tar`:

```
tar -tvf dist.tar
```

Copy the contents of the current directory to the floppy drive:

```
tar -cf /dev/fd0 .
```

Make an archive, `backup.tar`, of all the C source and header files in the current directory:

```
tar -cvf backup.tar *.[ch]
```

Files:

The controlling terminal (`/dev/tty`) is used to prompt the user for information when either or both the -i or -y options are specified.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

tcpdump

Dump traffic on a network

Syntax:

```
tcpdump [-AdDefKlLnNOpqRStuUvxX] [-c count] [-C file_size]
        [-E spi@ipaddr algo:secret,...] [-F file] [-G
rotate_seconds]
        [-i interface] [-m module] [-M secret] [-r file]
        [-s snaplen] [-T type] [-w file]
       [-W filecount] [-y datalinktype] [-z postrotate-command]

        [-Z user] [expression]
```

Runs on:

QNX Neutrino

Options:

-A

Print each packet (minus its link level header) in ASCII. Handy for capturing web pages.

-c *count*

Exit after receiving *count* packets.

-C *file_size*

Before writing a raw packet to a savefile, check whether the file is currently larger than *file_size* and, if so, close the current savefile and open a new one. Savefiles after the first savefile will have the name specified with the -w option, with a number after it, starting at 1 and continuing upward. The units of *file_size* are millions of bytes (1,000,000 bytes, not 1,048,576 bytes).

-d

Dump the compiled packet-matching code in a human readable form to standard output and stop.

-dd

Dump packet-matching code as a C program fragment.

-ddd

Copyright © 2014, QNX Software Systems Limited

Dump packet-matching code as decimal numbers (preceded with a count).

-D

Print the list of the network interfaces available on the system and on which `tcpdump` can capture packets. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. You can supply the interface name or the number to the -i option to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g. Windows systems, or UNIX systems lacking `ifconfig -a`); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string.

-e

Print the link-level header on each dump line.

-E *spi@ipaddr algo:secret,...*

Use *spi@ipaddr algo:secret* for decrypting IPsec ESP packets that are addressed to *addr* and contain Security Parameter Index value *spi*. You can specify additional combinations, separating them with commas or newlines.

Setting the secret for IPv4 ESP packets isn't currently supported.

The algorithm can be `des-cbc`, `3des-cbc`, `blowfish-cbc`, `rc3-cbc`, `cast128-cbc`, or `none`. The default is `des-cbc`.

The *secret* is the ASCII text for the ESP secret key. If preceded by `0x`, then a hexadecimal value is read.

The option assumes RFC 2406 ESP, not RFC 1827 ESP. The option is only for debugging purposes, and the use of this option with a true “secret” key is discouraged. By presenting the IPsec secret key onto command line, you make it visible to others, via `ps` and on other occasions.

In addition to the above syntax, you can use the syntax *file_name* to have `tcpdump` read the provided file. The file is opened on receiving the first ESP packet, so any special permissions that `tcpdump` may have been given should already have been given up.

-f

Print “foreign” IPv4 addresses numerically rather than symbolically.

Copyright © 2014, QNX Software Systems Limited

The test for “foreign” IPv4 addresses is done using the IPv4 address and netmask of the interface on which capture is being done. If that address or netmask isn't available, either because the interface on which capture is being done has no address or netmask, or because the capture is being done on the Linux “any” interface, which can capture on more than one interface, this option won't work correctly.

-F *file*

Use *file* as input for the filter expression. Any additional expression given on the command line is ignored.

-G *rotate_seconds*

If specified, rotates the dump file specified with the -w option every *rotate_seconds* seconds. Savefiles have the name specified by -w, which should include a time format as defined by *strftime()*. If no time format is specified, each new file overwrites the previous.

If used in conjunction with the -C option, file names take the form *file*<count>.

-i *interface*

Listen on *interface*. If unspecified, `tcpdump` searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match.

On Linux systems with 2.2 or later kernels, you can use an *interface* argument of `any` to capture packets from all interfaces. Note that captures on the `any` device aren't done in promiscuous mode.

You can use an interface number as printed by that option as the *interface* argument.

-K

Don't attempt to verify TCP checksums. This is useful for interfaces that perform the TCP checksum calculation in hardware; otherwise, all outgoing TCP checksums are flagged as bad.

-l

(“el”) Make *stdout* line buffered. This is useful if you want to see the data while capturing it. For example, `tcpdump -l | tee dat` or `tcpdump -l > dat & tail -f dat`.

-L

List the known data link types for the interface, and then exit.

Copyright © 2014, QNX Software Systems Limited

-m *module*

Load SMI MIB module definitions from file *module*. You can use this option several times to load several MIB modules into `tcpdump`.

-M *secret*

Use *secret* as a shared secret for validating the digests found in TCP segments with the TCP-MD5 option (RFC 2385), if present.

-n

Don't convert addresses (i.e., host addresses, port numbers, etc.) into names.

-N

Don't print domain name qualification of host names. For example, if you give this option, `tcpdump` prints `nic` instead of `nic.ddn.mil`.

-O

("Oh") Don't run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

-p

Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, you can't use -p as an abbreviation for `ether host {local-hw-addr} or ether broadcast`.

-q

Be quiet; print less protocol information, so that output lines are shorter.

-R

Assume ESP/AH packets to be based on old specification (RFC 1825 to RFC 1829). If specified, `tcpdump` doesn't print the replay prevention field. Since there is no protocol version field in ESP/AH specification, `tcpdump` can't deduce the version of ESP/AH protocol.

-r *file*

Read packets from *file* (which was created with the -w option). If *file* is `-`, `tcpdump` uses standard input.

-S

Print absolute, rather than relative, TCP sequence numbers.

-s *snaplen*

Copyright © 2014, QNX Software Systems Limited

Snarf *snaplen* bytes of data from each packet rather than the default of 68 (with SunOS's NIT, the minimum is actually 96). 68 bytes is adequate for IP, ICMP, TCP and UDP, but may truncate protocol information from name server and NFS packets (see below).

Packets truncated because of a limited snapshot are indicated in the output with `[|proto]`, where *proto* is the name of the protocol level at which the truncation has occurred.

Taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit *snaplen* to the smallest number that will capture the protocol information you're interested in. If you set *snaplen* to 0, `tcpdump` uses the required length to catch whole packets.

-T *type*

Force packets selected by *expression* to be interpreted as the specified *type*. Currently known types are:

- `aodv` — ad hoc On-demand Distance Vector protocol
- `cnfp` — Cisco NetFlow protocol
- `rpc` — Remote Procedure Call
- `rtp` — Real-Time Applications protocol
- `rtcp` — Real-Time Applications control protocol
- `snmp` — Simple Network Management Protocol
- `tftp` — Trivial File Transfer Protocol
- `vat` — Visual Audio Tool
- `wb` — distributed White Board

-t

Don't print a timestamp on each dump line.

-tt

Print an unformatted timestamp on each dump line.

-ttt

Print a delta (micro-second resolution) between current and previous line on each dump line.

-tttt

Print a timestamp in default format proceeded by date on each dump line.

Copyright © 2014, QNX Software Systems Limited

-ttttt

Print a delta (micro-second resolution) between current and first line on each dump line.

-u

Print undecoded NFS handles.

-U

Make output saved via the -w option packet-buffered; i.e. as each packet is saved, write it to the output file, rather than writing it only when the output buffer fills.

-v

When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. This option also enables additional packet integrity checks, such as verifying the IP and ICMP header checksum.

When writing to a file with the -w option, report, every 10 seconds, the number of packets captured.

-vv

Even more verbose output. For example, additional fields are printed from NFS reply packets, and SMB packets are fully decoded.

-vvv

Even more verbose output. For example, telnet SB ... SE options are printed in full. With -X, telnet options are printed in hexadecimal as well.

-w *file*

Write the raw packets to *file* rather than parsing and printing them. You can print them with the -r option. If *file* is `-`, `tcpdump` uses standard output.

-W *filecount*

Used in conjunction with the -C option, limit the number of files created to the specified number, and begin overwriting files from the beginning, thus creating a “rotating” buffer. In addition, it names the files with enough leading zeroes to support the maximum number of files, allowing you to sort them correctly.

Used in conjunction with the -G option, this option limits the number of rotated dump files that get created, exiting with a status of 0 when `tcpdump`

Copyright © 2014, QNX Software Systems Limited

reaches the limit. If you use it with -C as well, `tcpdump` uses cyclical files per timeslice.

-x

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link-level header) in hexadecimal. The smaller of the entire packet or *snaplen* bytes is printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Ethernet), the padding bytes are also printed when the higher layer packet is shorter than the required padding.

-xx

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link-level header, in hexadecimal.

-X

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link-level header) in hexadecimal and ASCII. This is very handy for analyzing new protocols.

-XX

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link-level header, in hexadecimal and ASCII.

-y datalinktype

Set the data link type to use while capturing packets to *datalinktype*.

-z postrotate-command

Used in conjunction with the -C or -G options, this makes `tcpdump` run *postrotate-command file*, where *file* is the savefile being closed after each rotation. For example, specifying -z gzip or -z bzip2 compresses each savefile using `gzip` or `bzip2`.

The `tcpdump` utility runs the command in parallel with the capture, using the lowest priority so that this doesn't disturb the capture process.

If you want to use a command that itself takes options or different arguments, write a shell script that takes the savefile name as the only argument, arrange the options and arguments as required, and then execute the command that you want.

Copyright © 2014, QNX Software Systems Limited

-Z user

Drop privileges (if `root`) and change the user ID to *user* and the group ID to the primary group of *user*.

Description:

The `tcpdump` utility prints a description of the contents of packets on a network interface that match the boolean *expression*. You can also run it with the -w option, which causes it to save the packet data to a file for later analysis, and/or with the -r option, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, `tcpdump` processes only those packets that match *expression*.

For information about the *expression* argument, see "*Expressions* (p. 1902)," below.

The `tcpdump` utility, if not run with the -c option, continues capturing packets until it's interrupted by a `SIGINT` signal (generated, for example, by typing your interrupt character, typically **Control**–**C**) or a `SIGTERM` signal (typically generated with the *`kill`* (p. 1026) command); if run with the -c option, `tcpdump` captures packets until it's interrupted by a `SIGINT` or `SIGTERM` signal, or the specified number of packets have been processed.

When `tcpdump` finishes capturing packets, it reports counts of:

- packets "captured" — the number of packets that `tcpdump` has received and processed.
- packets "received by filter" — the number of packets handed to the filter, not packets that passed the filter. This includes packets later dropped because there wasn't enough buffer space.
- packets "dropped by kernel" — the number of packets that were dropped, due to a lack of buffer space, by the packet-capture mechanism.

You must be `root` in order to read packets from a network interface. Reading a saved packet file doesn't require special privileges; you just need read permission for the file.

Expressions

The *expression* on the command line selects which packets to dump. If no *expression* is given, all packets on the net will be dumped. Otherwise, only packets for which *expression* is true are dumped.

The *expression* consists of one or more *primitives* that usually consist of an ID (name or number) preceded by one or more qualifiers. The different kinds of qualifier are:

type

Copyright © 2014, QNX Software Systems Limited

The kind of thing the ID name or number refers to. Possible types are `host`, `net`, `port`, and `portrange`. For example, `host xyz`, `net 128.3`, `port 20`, and `portrange 6000-6008`. If there's no type qualifier, `tcpdump` uses `host`.

dir

A particular transfer direction to and/or from the ID. Possible directions are `src`, `dst`, `src or dst`, and `src and dst`. For example, `src xyz`, `dst net 128.3`, `src or dst port ftp-data`. If there is no *dir* qualifier, **src or dst** is assumed.

For some link layers, such as SLIP and the "cooked" Linux capture mode used for the `any` device and for some other device types, you can use the `inbound` and `outbound` qualifiers to specify a desired direction.

proto

Restrict the match to a particular protocol. Possible protocols are `ether`, `fddi`, `tr`, `wlan`, `ip`, `ip6`, `arp`, `rarp`, `decnet`, `tcp`, and `udp`. For example, `ether src xyz`, `arp net 128.3`, `tcp port 21`, and `udp portrange 7000-7009`.

If there's no proto qualifier, all protocols consistent with the type are assumed. For example, `src xyz` means `(ip or arp or rarp) src xyz` (except the latter isn't legal syntax), `net abc` means `(ip or arp or rarp) net abc`, and `port 53` means `(tcp or udp) port 53`.

The `fddi` protocol is actually an alias for `ether`; the parser treats them identically as meaning "the data link level used on the specified network interface." FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. FDDI headers also contain other fields, but you can't name them explicitly in a filter expression.

Similarly, `tr` and `wlan` are aliases for `ether`; the previous paragraph's statements about FDDI headers also apply to Token Ring and 802.11 wireless LAN headers. For 802.11 headers, the destination address is the DA field and the source address is the SA field; the BSSID, RA, and TA fields aren't tested.

In addition to the above, there are some special "primitive" keywords that don't follow the pattern:

- `gateway`
- `broadcast`

Copyright © 2014, QNX Software Systems Limited

- `less`
- `greater`

and arithmetic expressions. All of these are described below.

You can build more complex filter expressions by using the words `and`, `or`, and `not` to combine primitives. For example, `host xyz and not port ftp and not port ftp-data`. To save typing, you can omit identical qualifier lists. For example, `tcp dst port ftp or ftp-data or domain` is exactly the same as `tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain`.

Allowable primitives are:

`dst host` ***host***

True if the IPv4/v6 destination field of the packet is *host*, which may be either an address or a name.

`src host` ***host***

True if the IPv4/v6 source field of the packet is *host*.

`host` ***host***

True if either the IPv4/v6 source or destination of the packet is *host*.

You can prepend any of the above host expressions with the keywords `ip`, `arp`, `rarp`, or `ip6` as in:

```
ip host host
```

which is equivalent to:

```
ether proto \ip and host host
```

If *host* is a name with multiple IP addresses, each address is checked for a match.

`ether dst` ***ehost***

True if the Ethernet destination address is *ehost*. The *ehost* may be either a name from `/etc/ethers` or a number (see *ethers*(3N) for numeric format).

`ether src` ***ehost***

True if the Ethernet source address is *ehost*.

`ether host` ***ehost***

True if either the Ethernet source or destination address is *ehost*.

`gateway` ***host***

Copyright © 2014, QNX Software Systems Limited

True if the packet used *host* as a gateway. That is, the Ethernet source or destination address was *host*, but neither the IP source nor the IP destination was *host*. The *host* must be a name and must be found both by the machine's host-name-to-IP-address resolution mechanisms (host name file, DNS, NIS, etc.) and by the machine's host-name-to-Ethernet-address resolution mechanism (/etc/ethers, etc.). (An equivalent expression is:

```
ether host ehost and not host host
```

which can be used with either names or numbers for *host / ehost*.) This syntax doesn't work in IPv6-enabled configuration at this moment.

dst net ***net***

True if the IPv4/v6 destination address of the packet has a network number of *net*. The *net* may be either a name from the networks database (/etc/networks, etc.) or a network number.

An IPv4 network number can be written as a dotted quad (e.g., 192.168.1.0), dotted triple (e.g., 192.168.1), dotted pair (e.g, 172.16), or single number (e.g., 10); the netmask is 255.255.255.255 for a dotted quad (which means that it's really a host match), 255.255.255.0 for a dotted triple, 255.255.0.0 for a dotted pair, or 255.0.0.0 for a single number.

An IPv6 network number must be written out fully; the netmask is ff:ff:ff:ff:ff:ff:ff:ff, so IPv6 “network” matches are really always host matches, and a network match requires a netmask length.

src net ***net***

True if the IPv4/v6 source address of the packet has a network number of *net*.

net ***net***

True if either the IPv4/v6 source or destination address of the packet has a network number of *net*.

net ***net*** **mask** ***netmask***

True if the IPv4 address matches *net* with the specific *netmask*. May be qualified with `src` or `dst`. Note that this syntax isn't valid for IPv6 *net*.

net ***net/len***

True if the IPv4/v6 address matches *net* with a netmask *len* bits wide. May be qualified with `src` or `dst`.

Copyright © 2014, QNX Software Systems Limited

dst port *port*

True if the packet is ip/tcp, ip/udp, ip6/tcp, or ip6/udp, and has a destination port value of *port*. The *port* can be a number or a name used in */etc/services* (p. 1744). (see tcp(4P) and udp(4P)).

If you use a name, both the port number and protocol are checked. If you use a number or ambiguous name, only the port number is checked (e.g., `dst port 513` will print both tcp/login traffic and udp/who traffic, and `port domain` will print both tcp/domain and udp/domain traffic).

src port *port*

True if the packet has a source port value of *port*.

port *port*

True if either the source or destination port of the packet is *port*.

dst portrange *port1-port2*

True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value between *port1* and *port2*. The *port1* and *port2* are interpreted in the same fashion as the *port* parameter for `port`.

src portrange *port1-port2*

True if the packet has a source port value between *port1* and *port2*.

portrange *port1-port2*

True if either the source or destination port of the packet is between *port1* and *port2*.

You can prepend any of the above port or port range expressions with the keywords `tcp` or `udp`, as in:

```
tcp src port port
```

which matches only tcp packets whose source port is *port*.

less *length*

True if the packet has a length less than or equal to *length*. This is equivalent to:

```
len <= length.
```

greater *length*

Copyright © 2014, QNX Software Systems Limited

True if the packet has a length greater than or equal to *length*. This is equivalent to:

```
len >= length.
```

ip proto ***protocol***

True if the packet is an IPv4 packet (see ip(4P)) of protocol type *protocol*. The *protocol* can be a number, or one of the names `icmp`, `icmp6`, `igmp`, `igrp`, `pim`, `ah`, `esp`, `vrrp`, `udp`, or `tcp`.

The identifiers `tcp`, `udp`, and `icmp` are also keywords and must be escaped via backslash (\), which is \\ in the C shell. This primitive doesn't chase the protocol header chain.

ip6 proto ***protocol***

True if the packet is an IPv6 packet of protocol type *protocol*. Note that this primitive doesn't chase the protocol header chain.

ip6 protochain ***protocol***

True if the packet is IPv6 packet, and contains a protocol header with the type *protocol* in its protocol header chain. For example

```
ip6 protochain 6
```

matches any IPv6 packet with TCP protocol header in the protocol header chain. The packet may contain, for example, authentication header, routing header, or hop-by-hop option header, between IPv6 header and TCP header. The BPF code emitted by this primitive is complex and can't be optimized by BPF optimizer code in `tcpdump`, so this can be somewhat slow.

ip protochain ***protocol***

Equivalent to `ip6 protochain` *protocol*, but this is for IPv4.

ether broadcast

True if the packet is an Ethernet broadcast packet. The `ether` keyword is optional.

ip broadcast

True if the packet is an IPv4 broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the subnet mask on the interface on which the capture is being done.

Copyright © 2014, QNX Software Systems Limited

If the subnet mask of the interface on which the capture is being done isn't available, either because the interface on which capture is being done has no netmask or because the capture is being done on the Linux “any” interface, which can capture on more than one interface, this check doesn't work correctly.

`ether multicast`

True if the packet is an Ethernet multicast packet. The `ether` keyword is optional. This is shorthand for `ether[0] & 1 != 0`.

`ip multicast`

True if the packet is an IPv4 multicast packet.

`ip6 multicast`

True if the packet is an IPv6 multicast packet.

`ether proto` ***protocol***

True if the packet is of ether type *protocol*. The *protocol* can be a number, or one of the names `ip`, `ip6`, `arp`, `rarp`, `atalk`, `aarp`, `decnet`, `sca`, `lat`, `mopdl`, `moprc`, `iso`, `stp`, `ipx`, or `netbeui`. Note these identifiers are also keywords and must be escaped via backslash (\).

In the case of FDDI (e.g. `fddi protocol arp`), Token Ring (e.g. `tr protocol arp`), and IEEE 802.11 wireless LANs (e.g. `wlan protocol arp`), for most of those protocols, the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI, Token Ring, or 802.11 header.

When filtering for most protocol identifiers on FDDI, Token Ring, or 802.11, `tcpdump` checks only the protocol ID field of an LLC header in so-called SNAP format with an Organizational Unit Identifier (OUI) of `0x000000`, for encapsulated Ethernet; it doesn't check whether the packet is in SNAP format with an OUI of `0x000000`. The exceptions are:

- `iso` — `tcpdump` checks the DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) fields of the LLC header
- `stp` and `netbeui` — `tcpdump` checks the DSAP of the LLC header
- `atalk` — `tcpdump` checks for a SNAP-format packet with an OUI of `0x080007` and the AppleTalk etype.

In the case of Ethernet, `tcpdump` checks the Ethernet type field for most of those protocols. The exceptions are:

- `iso`, `stp`, and `netbeui` — `tcpdump` checks for an 802.3 frame, and then checks the LLC header as it does for FDDI, Token Ring, and 802.11

Copyright © 2014, QNX Software Systems Limited

- `atalk` — `tcpdump` checks both for the AppleTalk etype in an Ethernet frame and for a SNAP-format packet as it does for FDDI, Token Ring, and 802.11
- `aarp` — `tcpdump` checks for the AppleTalk ARP etype in either an Ethernet frame or an 802.2 SNAP frame with an OUI of `0x000000`
- `ipx` — `tcpdump` checks for the IPX etype in an Ethernet frame, the IPX DSAP in the LLC header, the 802.3-with-no-LLC-header encapsulation of IPX, and the IPX etype in a SNAP frame.

`decnet src` ***host***

True if the DECNET source address is *host*, which is in the form `10.123`.

`decnet dst` ***host***

True if the DECNET destination address is *host*.

`decnet host` ***host***

True if either the DECNET source or destination address is *host*.

`ifname` ***interface***

True if the packet was logged as coming from the specified interface (applies only to packets logged by OpenBSD's *pf* (p. 1461)).

`on` ***interface***

Synonymous with the `ifname` modifier.

`rnr` ***num***

True if the packet was logged as matching the specified PF rule number (applies only to packets logged by OpenBSD's `pf`).

`rulenum` ***num***

Synonymous with the `rnr` modifier.

`reason` ***code***

True if the packet was logged with the specified PF reason code. The known codes are: `match`, `bad-offset`, `fragment`, `short`, `normalize`, and `memory` (applies only to packets logged by OpenBSD's `pf`).

`rset` ***name***

True if the packet was logged as matching the specified PF ruleset name of an anchored ruleset (applies only to packets logged by `pf`).

Copyright © 2014, QNX Software Systems Limited

ruleset *name*

Synonymous with the `rset` modifier.

srnr *num*

True if the packet was logged as matching the specified PF rule number of an anchored ruleset (applies only to packets logged by `pf`).

subrulenum *num*

Synonymous with the `srnr` modifier.

action *act*

True if PF took the specified action when the packet was logged. Known actions are `pass` and `block` (applies only to packets logged by OpenBSD's `pf`).

ip, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **iso**, **stp**, **ipx**, *netbeui*

Abbreviations for:

```
ether proto p
```

where *p* is one of the above protocols.

lat, **moprc**, **mopdl**

Abbreviations for:

```
ether proto p
```

where *p* is one of the above protocols. Note that `tcpdump` doesn't currently know how to parse these protocols.

type *wlan_type*

True if the IEEE 802.11 frame type matches the specified *wlan_type*. Valid *wlan_type*s are: `mgt`, `ctl`, and `data`.

type *wlan_type* **subtype** *wlan_subtype*

True if the IEEE 802.11 frame type matches the specified *wlan_type*, and frame subtype matches the specified *wlan_subtype*.

If the specified *wlan_type* is `mgt`, then valid *wlan_subtype*s are `assoc-req`, `assoc-resp`, `reassoc-req`, `reassoc-resp`, `probe-req`, `probe-resp`, `beacon`, `atim`, `disassoc`, `auth`, and `deauth`.

Copyright © 2014, QNX Software Systems Limited

If the specified *wlan_type* is `ctl`, then valid *wlan_subtype*s are `ps-poll`, `rts`, `cts`, `ack`, `cf-end`, and `cf-end-ack`.

If the specified *wlan_type* is `data`, then valid *wlan_subtype*s are `data`, `data-cf-ack`, `data-cf-poll`, `data-cf-ack-poll`, `null`, `cf-ack`, `cf-poll`, `cf-ack-poll`, `qos-data`, `qos-data-cf-ack`, `qos-data-cf-poll`, `qos-data-cf-ack-poll`, `qos`, `qos-cf-poll`, and `qos-cf-ack-poll`.

`subtype` ***wlan_subtype***

True if the IEEE 802.11 frame subtype matches the specified *wlan_subtype*, and frame has the type to which the specified *wlan_subtype* belongs.

`vlan` ***[vlan_id]***

True if the packet is an IEEE 802.1Q VLAN packet. If *[vlan_id]* is specified, this is true only if the packet has the specified *vlan_id*.

The first `vlan` keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a VLAN packet. You can use the `vlan`*[vlan_id]* expression more than once, to filter on VLAN hierarchies. Each use of that expression increments the filter offsets by 4.

For example:

```
vlan 100 && vlan 200
```

filters on VLAN 200 encapsulated within VLAN 100, and:

```
vlan && vlan 300 && ip
```

filters IPv4 protocols encapsulated in VLAN 300 encapsulated within any higher order VLAN.

`mpls` ***[label_num]***

True if the packet is an MPLS packet. If *[label_num]* is specified, this is true only if the packet has the specified *label_num*.

The first `mpls` keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a MPLS-encapsulated IP packet. You can use the `mpls` *[label_num]* expression more than once, to filter on MPLS hierarchies. Each use of that expression increments the filter offsets by 4.

Copyright © 2014, QNX Software Systems Limited

For example:

```
mpls 100000 && mpls 1024
```

filters packets with an outer label of 100000 and an inner label of 1024, and:

```
mpls && mpls 1024 && host 192.9.200.1
```

filters packets to or from 192.9.200.1 with an inner label of 1024 and any outer label.

pppoed

True if the packet is a PPP-over-Ethernet Discovery packet (Ethernet type 0x8863).

pppoes

True if the packet is a PPP-over-Ethernet Session packet (Ethernet type 0x8864).

The first `pppoes` keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a PPPoE session packet.

For example:

```
pppoes && ip
```

filters IPv4 protocols encapsulated in PPPoE.

tcp, udp, icmp

Abbreviations for:

```
ip proto p or ip6 proto p
```

where *p* is one of the above protocols.

iso proto *protocol*

True if the packet is an OSI packet of protocol type *protocol*. The *Protocol* can be a number, or one of the names `clnp`, `esis`, or `isis`.

clnp, esis, isis

Abbreviations for:

```
iso proto p
```


Copyright © 2014, QNX Software Systems Limited

where *p* is one of the above protocols.

l1, l2, iih, lsp, snp, csnp, psnp

Abbreviations for IS-IS PDU types.

vpi *n*

True if the packet is an ATM packet, for SunATM on Solaris, with a virtual path identifier of *n*.

vci *n*

True if the packet is an ATM packet, for SunATM on Solaris, with a virtual channel identifier of *n*.

lane

True if the packet is an ATM packet, for SunATM on Solaris, and is an ATM LANE packet.

The first `lane` keyword encountered in *expression* changes the tests done in the remainder of *expression* on the assumption that the packet is either a LANE emulated Ethernet packet or a LANE LE Control packet. If `lane` isn't specified, the tests are done under the assumption that the packet is an LLC-encapsulated packet.

llc

True if the packet is an ATM packet, for SunATM on Solaris, and is an LLC-encapsulated packet.

oamf4s

True if the packet is an ATM packet, for SunATM on Solaris, and is a segment OAM F4 flow cell (VPI=0 & VCI=3).

oamf4e

True if the packet is an ATM packet, for SunATM on Solaris, and is an end-to-end OAM F4 flow cell (VPI=0 & VCI=4).

oamf4

True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

oam

Copyright © 2014, QNX Software Systems Limited

True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

metac

True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit (VPI=0 & VCI=1).

bcc

True if the packet is an ATM packet, for SunATM on Solaris, and is on a broadcast signaling circuit (VPI=0 & VCI=2).

sc

True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit (VPI=0 & VCI=5).

ilmic

True if the packet is an ATM packet, for SunATM on Solaris, and is on an ILMI circuit (VPI=0 & VCI=16).

connectmsg

True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Connect Ack, Release, or Release Done message.

metaconnect

True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Release, or Release Done message.

expr relop expr

True if the relation holds, where *relop* is one of >, <, >=, <=, =, or `!=`, and *expr* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators (`+`, `-`, `*`, `/`, `&`, `|`, `<<`, `>>`), a length operator, and special packet data accessors. Note that all comparisons are unsigned, so that, for example, `0x80000000` and `0xffffffff` are greater than 0. To access data inside the packet, use the following syntax:

```
proto [expr : size ]
```

where *proto* is one of `ether`, `fddi`, `tr`, `wlan`, `ppp`, `slip`, `link`, `ip`, `arp`, `rarp`, `tcp`, `udp`, `icmp`, `ip6`, or `radio`, and indicates the protocol layer for the index operation (`ether`, `fddi`, `wlan`, `tr`, `ppp`, `slip`, and `link` all

Copyright © 2014, QNX Software Systems Limited

refer to the link layer; `radio` refers to the "radio header" added to some 802.11 captures).

The `tcp`, `udp`, and other upper-layer protocol types apply only to IPv4, not IPv6 (this will be fixed in the future).

The byte offset, relative to the indicated protocol layer, is given by *expr*. The *size* is optional and indicates the number of bytes in the field of interest; it can be one, two, or four, and defaults to one. The length operator, indicated by the keyword `len`, gives the length of the packet.

For example, `ether[0] & 1 != 0` catches all multicast traffic. The expression `ip[0] & 0xf != 5` catches all IPv4 packets with options. The expression `ip[6:2] & 0x1fff = 0` catches only unfragmented IPv4 datagrams and frag zero of fragmented IPv4 datagrams. This check is implicitly applied to the `tcp` and `udp` index operations. For instance, `tcp[0]` always means the first byte of the TCP *header*, and never means the first byte of an intervening fragment.

Some offsets and field values may be expressed as names rather than as numeric values. The following protocol header field offsets are available: `icmptype` (ICMP type field), `icmpcode` (ICMP code field), and `tcpflags` (TCP flags field).

The following ICMP type field values are available: `icmp-echoreply`, `icmp-unreach`, `icmp-sourcequench`, `icmp-redirect`, `icmp-echo`, `icmp-routeradvert`, `icmp-routersolicit`, `icmp-timxceed`, `icmp-paramprob`, `icmp-tstamp`, `icmp-tstampreply`, `icmp-ireq`, `icmp-ireqreply`, `icmp-maskreq`, and `icmp-maskreply`.

The following TCP flags field values are available: `tcp-fin`, `tcp-syn`, `tcp-rst`, `tcp-push`, `tcp-ack`, `tcp-urg`.

You can combine primitives by using:

- a parenthesized group of primitives and operators (parentheses are special to the shell and must be escaped)
- negation: `!` or `not`
- concatenation: && or `and`
- alternation: `||` or `or`

Negation has highest precedence. Alternation and concatenation have equal precedence and associate from left to right. Note that explicit `and` tokens, not juxtaposition, are now required for concatenation.

Copyright © 2014, QNX Software Systems Limited

If an identifier is given without a keyword, the most recent keyword is assumed. For example:

```
not host vs and ace
```

is short for:

```
not host vs and host ace
```

which shouldn't be confused with:

```
not ( host vs or ace )
```

You can pass expression arguments to `tcpdump` as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains shell metacharacters, it's easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

Output format

The output of `tcpdump` is protocol-dependent. The following gives a brief description and examples of most of the formats.

Link-level headers

If you specify the -e option, the link-level header is printed out. On Ethernets, the source and destination addresses, protocol, and packet length are printed.

On FDDI networks, the -e option causes `tcpdump` to print the frame-control field, the source and destination addresses, and the packet length. The frame-control field governs the interpretation of the rest of the packet. Normal packets (such as those containing IP datagrams) are “async” packets, with a priority value between 0 and 7; for example, `async4`. Such packets are assumed to contain an 802.2 Logical Link Control (LLC) packet; the LLC header is printed if it is *not* an ISO datagram or a so-called SNAP packet.

On Token Ring networks, the -e option causes `tcpdump` to print the access-control and frame-control fields, the source and destination addresses, and the packet length. As on FDDI networks, packets are assumed to contain an LLC packet. Regardless of whether the -e option is specified or not, the source routing information is printed for source-routed packets.

On 802.11 networks, the -e option causes `tcpdump` to print the frame-control fields, all of the addresses in the 802.11 header, and the packet length. As on FDDI networks, packets are assumed to contain an LLC packet.

The following description assumes familiarity with the SLIP compression algorithm described in RFC 1144.

On SLIP links, a direction indicator (`I` for inbound, `O` for outbound), packet type, and compression information are printed out. The packet type is printed first. The three

Copyright © 2014, QNX Software Systems Limited

types are `ip`, `utcp`, and `ctcp`. No further link information is printed for `ip` packets. For TCP packets, the connection identifier is printed following the type. If the packet is compressed, its encoded header is printed out. The special cases are printed out as *S+*n* and *SA+*n*, where *n* is the amount by which the sequence number (or sequence number and ack) has changed. If it isn't a special case, zero or more changes are printed. A change is indicated by U (urgent pointer), W (window), A (ack), S (sequence number), and I (packet ID), followed by a delta (+n or -n), or a new value (=n). Finally, the amount of data in the packet and compressed header length are printed.

For example, the following line shows an outbound compressed TCP packet, with an implicit connection identifier; the ack has changed by 6, the sequence number by 49, and the packet ID by 6; there are 3 bytes of data and 6 bytes of compressed header:

```
O ctcp * A+6 S+49 I+6 3 (6)
```

ARP/RARP packets

Arp/rarp output shows the type of request and its arguments. The format is intended to be self-explanatory. Here is a short sample taken from the start of an `rlogin` from host *rtsg* to host *csam*:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

The first line says that rtsg sent an arp packet asking for the Ethernet address of Internet host csam. Csam replies with its Ethernet address (in this example, Ethernet addresses are in uppercase, and Internet addresses are in lowercase).

This would look less redundant if we had done *tcpdump -n*:

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

If we had done *tcpdump -e*, the fact that the first packet is broadcast and the second is point-to-point would be visible:

```
RTSG Broadcast 0806  64: arp who-has csam tell rtsg
CSAM RTSG 0806  64: arp reply csam is-at CSAM
```

For the first packet this says the Ethernet source address is RTSG, the destination is the Ethernet broadcast address, the type field contained hexadecimal 0806 (type ETHER_ARP) and the total length was 64 bytes.

TCP Packets

The following description assumes familiarity with the TCP protocol described in RFC 793. If you aren't familiar with the protocol, neither this description nor `tcpdump` will be of much use to you.

Copyright © 2014, QNX Software Systems Limited

The general format of a tcp protocol line is:

```
src > dst: flags data-seqno ack window urgent options
```

Src and *dst* are the source and destination IP addresses and ports. *Flags* are some combination of S (SYN), F (FIN), P (PUSH), R (RST), W (ECN CWR) or E (ECN-Echo), or a single `.` (no flags). *Data-seqno* describes the portion of sequence space covered by the data in this packet (see example below). *Ack* is sequence number of the next data expected the other direction on this connection. *Window* is the number of bytes of receive buffer space available the other direction on this connection. *Urg* indicates there is urgent data in the packet. *Options* are tcp options enclosed in angle brackets (e.g., <mss 1024>).

The *src*, *dst*, and *flags* are always present. The other fields depend on the contents of the packet's tcp protocol header and are output only if appropriate.

Here's the opening portion of an rlogin from host `rtsg` to host `csam`:

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The first line says that tcp port 1023 on rtsg sent a packet to port `login` on csam. The `S` indicates that the `SYN` flag was set. The packet sequence number was 768512 and it contained no data. (The notation is *first*:*last*(*nbytes*), which means “sequence numbers *first* up to but not including *last* which is *nbytes* bytes of user data.”) There was no piggy-backed ack, the available receive window was 4096 bytes and there was a max-segment-size option requesting an mss of 1024 bytes.

Csam replies with a similar packet except it includes a piggy-backed ack for rtsg's SYN. Rtsg then acks csam's SYN. The `.` means no flags were set. The packet contained no data so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time `tcpdump` sees a TCP “conversation”, it prints the sequence number from the packet. On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being 1). The -S option overrides this feature, causing the original sequence numbers to be output.

On the sixth line, rtsg sends csam 19 bytes of data (bytes 2 through 20 in the rtsg -> csam side of the conversation). The PUSH flag is set in the packet. On the seventh line, csam says it's received data sent by rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer since csam's receive window has gotten 19 bytes smaller. Csam also sends one byte of data to rtsg in this packet. On the eighth and ninth lines, csam sends two bytes of urgent, pushed data to rtsg.

Copyright © 2014, QNX Software Systems Limited

If the snapshot was small enough that `tcpdump` didn't capture the full TCP header, it interprets as much of the header as it can and then reports "[|*tcp*]" to indicate the remainder couldn't be interpreted. If the header contains a bogus option (one with a length that's either too small or beyond the end of the header), `tcpdump` reports it as "[*bad opt*]" and doesn't interpret any further options (since it's impossible to tell where they start). If the header length indicates options are present but the IP datagram length isn't long enough for the options to actually be there, `tcpdump` reports it as "[*bad hdr length*]".

Capturing TCP packets with particular flag combinations (SYN-ACK, URG-ACK, etc.)

There are 8 bits in the control bits section of the TCP header:

```
CWR | ECE | URG | ACK | PSH | RST | SYN | FIN
```

Let's assume that we want to watch packets used in establishing a TCP connection. Recall that TCP uses a 3-way handshake protocol when it initializes a new connection; the connection sequence with regard to the TCP control bits is

1. Caller sends SYN.
2. Recipient responds with SYN, ACK.
3. Caller sends ACK.

Now we're interested in capturing packets that have only the SYN bit set (Step 1). Note that we don't want packets from step 2 (SYN-ACK), just a plain initial SYN. What we need is a correct filter expression for `tcpdump`.

Recall the structure of a TCP header without options:

```
 0                            15                              31
-----------------------------------------------------------------
|          source port          |       destination port        |
-----------------------------------------------------------------
|                        sequence number                        |
-----------------------------------------------------------------
|                     acknowledgment number                     |
-----------------------------------------------------------------
|  HL   | rsvd  |C|E|U|A|P|R|S|F|        window size            |
-----------------------------------------------------------------
|         TCP checksum          |       urgent pointer          |
-----------------------------------------------------------------
```

A TCP header usually holds 20 octets of data, unless options are present. The first line of the graph contains octets 0 - 3, the second line shows octets 4 - 7 etc.

Starting to count with 0, the relevant TCP control bits are contained in octet 13:

```
 0             7|             15|             23|             31
----------------|---------------|---------------|----------------
|  HL   | rsvd  |C|E|U|A|P|R|S|F|        window size            |
----------------|---------------|---------------|----------------
|               |  13th octet   |               |               |
```

Let's have a closer look at the thirteenth octet:

```
                |               |
                |---------------|
                |C|E|U|A|P|R|S|F|
                |---------------|
                |7   5   3     0|
```

Copyright © 2014, QNX Software Systems Limited

These are the TCP control bits we're interested in. We have numbered the bits in this octet from 0 to 7, right to left, so the PSH bit is bit number 3, while the URG bit is number 5.

Recall that we want to capture packets with only SYN set. Let's see what happens to octet 13 if a TCP datagram arrives with the SYN bit set in its header:

```
|C|E|U|A|P|R|S|F|
|---------------|
|0 0 0 0 0 0 1 0|
|---------------|
|7 6 5 4 3 2 1 0|
```

Looking at the control bits section, we see that only bit number 1 (SYN) is set.

Assuming that octet number 13 is an 8-bit unsigned integer in network byte order, the binary value of this octet is `00000010`, and its decimal representation is:

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 1*2 + 0*2  =  2
```

We're almost done, because now we know that if only SYN is set, the value of the thirteenth octet in the TCP header, when interpreted as a 8-bit unsigned integer in network byte order, must be exactly 2.

This relationship can be expressed as:

```
tcp[13] == 2
```

We can use this expression as the filter for `tcpdump` in order to watch packets which have only SYN set:

```
tcpdump -i xl0 tcp[13] == 2
```

The expression says “let the thirteenth octet of a TCP datagram have the decimal value 2”, which is exactly what we want.

Now, let's assume that we need to capture SYN packets, but we don't care if ACK or any other TCP control bit is set at the same time. Let's see what happens to octet 13 when a TCP datagram with SYN-ACK set arrives:

```
|C|E|U|A|P|R|S|F|
|---------------|
|0 0 0 1 0 0 1 0|
|---------------|
|7 6 5 4 3 2 1 0|
```

Now bits 1 and 4 are set in the thirteenth octet. The binary value of octet 13 is `00010010`, which translates to decimal:

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 1*2 + 0*2 + 0*2 + 1*2 + 0*2   = 18
```

Now we can't just use 'tcp[13] == 18' in the `tcpdump` filter expression, because that would select only those packets that have SYN-ACK set, but not those with only SYN set. Remember that we don't care if ACK or any other control bit is set as long as SYN is set.

Copyright © 2014, QNX Software Systems Limited

In order to achieve our goal, we need to logically AND the binary value of octet 13 with some other value to preserve the SYN bit. We know that we want SYN to be set in any case, so we'll logically AND the value in the thirteenth octet with the binary value of a SYN:

```
     00010010 SYN-ACK              00000010 SYN
AND  00000010 (we want SYN)   AND  00000010 (we want SYN)
     --------                      --------
=    00000010                 =    00000010
```

We see that this AND operation delivers the same result regardless whether ACK or another TCP control bit is set. The decimal representation of the AND value as well as the result of this operation is 2 (binary 00000010), so we know that for packets with SYN set the following relation must hold true:

```
( ( value of octet 13 ) AND ( 2 ) ) == ( 2 )
```

This points us to the `tcpdump` filter expression

```
tcpdump -i xl0 'tcp[13] & 2 == 2'
```

Note that you should use single quotes or a backslash in the expression to hide the AND ('&') special character from the shell.

UDP Packets

UDP format is illustrated by this rwho packet:

```
actinide.who > broadcast.who: udp 84
```

This says that port *who* on host *actinide* sent a udp datagram to port *who* on host *broadcast*, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information printed. In particular, Domain Name service requests (RFCs 1034 and 1035) and Sun RPC calls (RFC 1050) to NFS.

UDP Name Server Requests

The following description assumes familiarity with the Domain Service protocol described in RFC 1035.

Name server requests are formatted as:

```
src > dst: id op? flags qtype qclass name (len)
```

```
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

Host `h2opolo` asked the domain server on `helios` for an address record (qtype=A) associated with the name *ucbvax.berkeley.edu.* The query ID was 3. The + indicates the *recursion desired* flag was set. The query length was 37 bytes, not including the UDP and IP protocol headers. The query operation was the normal one, *Query*, so the op field was omitted. If the op had been anything else, it would have been printed

Copyright © 2014, QNX Software Systems Limited

between the 3 and the +. Similarly, the qclass was the normal one, *C_IN*, and omitted. Any other qclass would have been printed immediately after the A.

A few anomalies are checked and may result in extra fields enclosed in square brackets: If a query contains an answer, authority records or additional records section, *ancount*, *nscount*, or *arcount* are printed as [*n*a], [*n*n], or [*n*au], where *n* is the appropriate count. If any of the response bits are set (AA, RA or rcode) or any of the "must be zero" bits are set in bytes two and three, [b2&3=*x*] is printed, where *x* is the hexadecimal value of header bytes two and three.

UDP Name Server Responses

Name server responses are formatted as

```
src > dst: id op rcode flags a/n/au type class data (len)

helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, helios responds to query ID 3 from h2opolo with 3 answer records, 3 name server records and 7 additional records. The first answer record is type A (address) and its data is Internet address 128.32.137.3. The total size of the response was 273 bytes, excluding UDP and IP headers. The op (Query) and response code (NoError) were omitted, as was the class (C_IN) of the A record.

In the second example, helios responds to query 2 with a response code of non-existent domain (NXDomain) with no answers, one name server and no authority records. The * indicates that the *authoritative answer* bit was set. Since there were no answers, no type, class or data were printed.

Other flag characters that might appear are - (recursion available, RA, *not* set) and | (truncated message, TC, set). If the question section doesn't contain exactly one entry, [*n*q] is printed.

Note that name server requests and responses tend to be large and the default *snaplen* of 68 bytes may not capture enough of the packet to print. Use the -s flag to increase the snaplen if you need to seriously investigate name server traffic. For example, -s 128.

SMB/CIFS decoding

tcpdump now includes fairly extensive SMB/CIFS/NBT decoding for data on UDP/137, UDP/138 and TCP/139. Some primitive decoding of IPX and NetBEUI SMB data is also done.

By default, a fairly minimal decode is done, with a much more detailed decode done if -v is used. Be warned that with -v a single SMB packet may take up a page or more, so only use -v if you really want all the details.

NFS Requests and Replies

Copyright © 2014, QNX Software Systems Limited

Sun NFS (Network File System) requests and replies are printed as:

```
src.xid > dst.nfs: len op args
src.nfs > dst.xid: reply stat len op results
```

For example:

```
sushi.6709 > wrl.nfs: 112 readlink fh 21,24/10.73165
wrl.nfs > sushi.6709: reply ok 40 readlink "../var"
sushi.201b > wrl.nfs:
        144 lookup fh 9,74/4096.6878 "xcolors"
wrl.nfs > sushi.201b:
        reply ok 128 lookup fh 9,74/4134.3150
```

In the first line, host `sushi` sends a transaction with ID 6709 to `wrl` (note that the number following the source host is a transaction ID, *not* the source port). The request was 112 bytes, excluding the UDP and IP headers. The operation was a `readlink` (read symbolic link) on file handle (`fh`) 21,24/10.731657119. (If you're lucky, as in this case, the file handle can be interpreted as a major, minor device number pair, followed by the inode number and generation number.) The `wrl` host replies `ok` with the contents of the link.

In the third line, `sushi` asks `wrl` to look up the name `xcolors` in directory file 9,74/4096.6878. Note that the data printed depends on the operation type. The format is intended to be self explanatory if read in conjunction with an NFS protocol spec.

If you specify the -v (verbose) option, additional information is printed. For example:

```
sushi.1372a > wrl.nfs:
        148 read fh 21,11/12.195 8192 bytes @ 24576
wrl.nfs > sushi.1372a:
        reply ok 1472 read REG 100664 ids 417/0 sz 29388
```

(The -v optioin also prints the IP header TTL, ID, length, and fragmentation fields, which have been omitted from this example.) In the first line, `sushi` asks `wrl` to read 8192 bytes from file 21,11/12.195, at byte offset 24576. The `wrl` host replies `ok`; the packet shown on the second line is the first fragment of the reply, and hence is only 1472 bytes long (the other bytes will follow in subsequent fragments, but these fragments don't have NFS or even UDP headers and so might not be printed, depending on the filter expression used). Because the -v flag is given, some of the file attributes (which are returned in addition to the file data) are printed: the file type ("REG", for regular file), the file mode (in octal), the uid and gid, and the file size.

If you specify more than one -v option, even more details are printed.

Note that NFS requests are very large and much of the detail won't be printed unless *snaplen* is increased. Try using -s 192 to watch NFS traffic.

NFS reply packets don't explicitly identify the RPC operation. Instead, `tcpdump` keeps track of "recent" requests, and matches them to the replies using the transaction ID. If a reply doesn't closely follow the corresponding request, it might not be parsable.

AFS Requests and Replies

Copyright © 2014, QNX Software Systems Limited

Transarc AFS (Andrew File System) requests and replies are printed as:

```
src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call call-name args
src.sport > dst.dport: rx packet-type service reply call-name args

elvis.7001 > pike.afsfs:
        rx data fs call rename old fid 536876964/1/1 ".newsrc.new"
        new fid 536876964/1/1 ".newsrc"
pike.afsfs > elvis.7001: rx data fs reply rename
```

In the first line, host elvis sends a RX packet to pike. This was a RX data packet to the fs (fileserver) service, and is the start of an RPC call. The RPC call was a rename, with the old directory file ID of 536876964/1/1 and an old filename of `.newsrc.new`, and a new directory file ID of 536876964/1/1 and a new filename of `.newsrc`. The host pike responds with a RPC reply to the rename call (which was successful, because it was a data packet and not an abort packet).

In general, all AFS RPCs are decoded at least by RPC call name. Most AFS RPCs have at least some of the arguments decoded (generally only the "interesting" arguments, for some definition of interesting).

The format is intended to be self-describing, but it will probably not be useful to people who aren't familiar with the workings of AFS and RX.

If the -v (verbose) flag is given twice, acknowledgment packets and additional header information is printed, such as the the RX call ID, call number, sequence number, serial number, and the RX packet flags.

If the -v flag is given twice, additional information is printed, such as the the RX call ID, serial number, and the RX packet flags. The MTU negotiation information is also printed from RX ack packets.

If you specify the -v option three times, the security index and service ID are printed.

Error codes are printed for abort packets, with the exception of Ubik beacon packets (because abort packets are used to signify a yes vote for the Ubik protocol).

Note that AFS requests are very large and many of the arguments won't be printed unless *snaplen* is increased. Try using -s 256 to watch AFS traffic.

AFS reply packets don't explicitly identify the RPC operation. Instead, `tcpdump` keeps track of "recent" requests, and matches them to the replies using the call number and service ID. If a reply doesn't closely follow the corresponding request, it might not be parsable.

KIP AppleTalk (DDP in UDP)

AppleTalk DDP packets encapsulated in UDP datagrams are de-encapsulated and dumped as DDP packets (i.e., all the UDP header information is discarded). The file `/etc/atalk.names` is used to translate AppleTalk net and node numbers to names. Lines in this file have the form:

```
number  name
```


Copyright © 2014, QNX Software Systems Limited

```
1.254           ether
16.1            icsd-net
1.254.110       ace
```

The first two lines give the names of AppleTalk networks. The third line gives the name of a particular host. (A host is distinguished from a net by the third octet in the number; a net number *must* have two octets and a host number *must* have three octets.) The number and name should be separated by whitespace (blanks or tabs). The `/etc/atalk.names` file may contain blank lines or comment lines (lines starting with a #).

AppleTalk addresses are printed in the form:

```
net.host.port
```

```
144.1.209.2 > icsd-net.112.220
office.2 > icsd-net.112.220
jssmag.149.235 > icsd-net.2
```

(If the */etc/atalk.names* doesn't exist or doesn't contain an entry for some AppleTalk host/net number, addresses are printed in numeric form.) In the first example, NBP (DDP port 2) on net 144.1 node 209 is sending to whatever is listening on port 220 of net icsd node 112. The second line is the same except the full name of the source node is known (`office`). The third line is a send from port 235 on net `jssmag` node 149 to broadcast on the icsd-net NBP port (note that the broadcast address (255) is indicated by a net name with no host number; for this reason it's a good idea to keep node names and net names distinct in `/etc/atalk.names`).

NBP (name binding protocol) and ATP (AppleTalk transaction protocol) packets have their contents interpreted. Other protocols just dump the protocol name (or number if no name is registered for the protocol) and packet size.

NBP packets are formatted like the following examples:

```
icsd-net.112.220 > jssmag.2: nbp-lkup 190: “=:LaserWriter@*”
jssmag.209.2 > icsd-net.112.220: nbp-reply 190: “RM1140:LaserWriter@*” 250
techpit.2 > icsd-net.112.220: nbp-reply 190: “techpit:LaserWriter@*” 186
```

The first line is a name-lookup request for laserwriters sent by net icsd host 112 and broadcast on net `jssmag`. The nbp ID for the lookup is 190. The second line shows a reply for this request (note that it has the same id) from host `jssmag.209` saying that it has a laserwriter resource named `RM1140` registered on port 250. The third line is another reply to the same request saying host techpit has laserwriter `techpit` registered on port 186.

ATP packet formatting is demonstrated by the following example:

```
jssmag.209.165 > helios.132: atp-req  12266<0-7> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:0 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:1 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:2 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:4 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:6 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp*12266:7 (512) 0xae040000
jssmag.209.165 > helios.132: atp-req  12266<3,5> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
jssmag.209.165 > helios.132: atp-rel  12266<0-7> 0xae030001
jssmag.209.133 > helios.132: atp-req* 12267<0-7> 0xae030002
```

Copyright © 2014, QNX Software Systems Limited

The `jssmag.209` host initiates transaction ID 12266 with host `helios` by requesting up to 8 packets (the `<0-7>`). The hexadecimal number at the end of the line is the value of the *userdata* field in the request.

The `helios` host responds with 8 512-byte packets. The `:digit` following the transaction ID gives the packet sequence number in the transaction and the number in parentheses is the amount of data in the packet, excluding the ATP header. The * on packet 7 indicates that the EOM bit was set.

The `jssmag.209` host then requests that packets 3 and 5 be retransmitted; `helios` resends them, and then `jssmag.209` releases the transaction. Finally, `jssmag.209` initiates the next request. The * on the request indicates that XO ("exactly once") wasn't set.

IP Fragmentation

Fragmented Internet datagrams are printed as:

```
(frag id:size@offset+)
(frag id:size@offset)
```

The first form indicates there are more fragments. The second indicates this is the last fragment.

Id is the fragment id. *Size* is the fragment size (in bytes) excluding the IP header. *Offset* is this fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the frag info is printed after the protocol info. Fragments after the first contain no higher level protocol header and the frag info is printed after the source and destination addresses. For example, here is part of an ftp from arizona.edu to lbl-rtsg.arpa over a CSNET connection that doesn't appear to handle 576 byte datagrams:

```
arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096 (frag 595a:328@0+)
arizona > rtsg: (frag 595a:204@328)
rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

There are a couple of things to note here: First, addresses in the second line don't include port numbers. This is because the TCP protocol information is all in the first fragment and we have no idea what the port or sequence numbers are when we print the later fragments. Second, the tcp sequence information in the first line is printed as if there were 308 bytes of user data when, in fact, there are 512 bytes (308 in the first frag and 204 in the second). If you are looking for holes in the sequence space or trying to match up acks with packets, this can fool you.

A packet with the IP *don't fragment* flag is marked with a trailing `(DF)`.

Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time, in the form *hh:mm:ss.frac* and is as accurate as the kernel's clock. The timestamp reflects the time `io-pkt` first saw the packet. No attempt is made to

Copyright © 2014, QNX Software Systems Limited

account for the time lag between when the Ethernet interface removed the packet from the wire and when `io-pkt` serviced the "new packet" interrupt.

Examples:

Print all packets arriving at or departing from `sundown`:

```
tcpdump host sundown
```

Print traffic between `helios` and either `hot` or `ace`:

```
tcpdump host helios and \( hot or ace \)
```

Print all IP packets between `ace` and any host except `helios`:

```
tcpdump ip host ace and not helios
```

Print all traffic between local hosts and hosts at Berkeley:

```
tcpdump net ucb-ether
```

Print all ftp traffic through Internet gateway `snup` (note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):

```
tcpdump 'gateway snup and (port ftp or ftp-data)'
```

Print traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net):

```
tcpdump ip and not net localnet
```

Print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host:

```
tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net localnet'
```

Print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets (IPv6 is left as an exercise for the reader):

```
tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
```

Print IP packets longer than 576 bytes sent through gateway `snup`:

```
tcpdump 'gateway snup and ip[2:2] > 576'
```

Print IP broadcast or multicast packets that weren't sent via Ethernet broadcast or multicast:

```
tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'
```

Print all ICMP packets that aren't echo requests/replies (i.e., not ping packets):

```
tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

Successful completion.

1

An error occurred.

Contributing author:

The original authors are Van Jacobson, Craig Leres, and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA. The `tcpdump` utility is currently maintained by `tcpdump.org`.

IPv6/IPsec support is added by WIDE/KAME project. This program uses Eric Young's SSLeay library, under specific configuration.

Copyright © 2014, QNX Software Systems Limited

tee

Duplicate standard input (POSIX)

Syntax:

```
tee [-ai] [file... ]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Append output to the specified file rather than overwriting it.

-i

Ignore the SIGINT signal.

file

A pathname of an output file. If you don't specify any files, tee writes to the standard output.

Description:

The tee utility copies standard input to standard output, making a copy in zero or more files. It doesn't buffer its output. You usually use tee in a pipeline, in order to make a copy of the output of a utility.

If -a is specified, the named file is opened for appending and data from tee is added to the file's existing data. If -a isn't specified, the file is opened for writing and the original data in the file is lost.

While it normally takes the default action for all signals, tee ignores SIGINT if -i is specified.

Examples:

Look for some spots in a collection of C programs where a variable *count* looks like it's being assigned a value, and tee the output both to standard output (where you can see it immediately) and a file output where you can look at it later (note the grep

Copyright © 2014, QNX Software Systems Limited

example here isn't sufficient for an exhaustive examination of where *count* might be assigned a value):

```
grep 'count *[^+-/\*]\{,1\}= *[!-([:alnum:]]' | tee output
```

Files:

An output file is created for each *file* operand.

Exit status:

0

Success.

> 0

An error occurred.

Errors:

If a write to any successfully opened file fails, writes to standard output and other successfully opened files continue. The exit status, however, is nonzero.

Copyright © 2014, QNX Software Systems Limited

telnet

User interface to the TELNET protocol (UNIX)

Syntax:

```
telnet [-8] [-c] [-d] [-E] [-e escape_char]
       [-L] [-N] [-n tracefile] [-P policy]
       [-S tos] [host [port]]
```

Runs on:

QNX Neutrino

Options:

-8

Allow an eight-bit input data path at all times. Without this option, parity bits are stripped whenever the remote side's stop and start characters are ^S and ^Q.

-c

Disables the reading of the user's `.telnetrc` file. (See the skiprc argument of the `toggle` telnet command, below.)

-d

Set the initial value of the debug toggle to TRUE.

-E

Disable the `telnet` escape character.

-e *escape_char*

Set the initial `telnet` escape character to *escape_char* (default is **Ctrl**–**]**). This character lets you switch to `telnet`'s command mode.

-L

Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-N

Numeric host address. Prevents look up of a symbolic name when the destination host is given as an IP address.

Copyright © 2014, QNX Software Systems Limited

-n ***tracefile***

Record trace information in the specified file. See the set *tracefile* command, below.

-P ***policy***

Use the IPsec policy specification string *policy* for the connections. For details of the IPsec policy control, see the QNX Neutrino *C Library Reference*.

-S ***tos***

Sets the IP type-of-service (TOS) option for the telnet connection to the value *tos*, which can be a numeric TOS value or, on systems that support it, a symbolic TOS name found in the `/etc/iptos` file.

host

The official name, an alias, or the Internet address of a remote host.

port

A port number (address of an application). If a number isn't specified, the default `telnet` port is used.

Description:

Use the `telnet` command to communicate with another host via the TELNET protocol. If you invoke `telnet` without a *host* argument, it enters command mode and displays this prompt:

```
telnet>
```

In command mode, `telnet` accepts and executes the commands listed in "*Telnet commands* (p. 1933)," below.

If you invoke `telnet` with a *host* argument, it opens a connection to the named host (i.e. it performs an `open` command).

If LINEMODE is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system relays that information. The remote system also relays changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In "character-at-a-time" mode, most text that's typed is immediately sent to the remote host for processing.

In "old-line-by-line" mode, all text is echoed locally, and (in most cases) only completed lines are sent to the remote host. You can use the "local echo character" (initially

Copyright © 2014, QNX Software Systems Limited

Ctrl–**E**) to turn off and on the local echo (you use this mostly to enter passwords without echoing).

If the LINEMODE option is enabled, or if the `localchars` toggle is TRUE (the default for "old-line-by-line"), your `quit`, `intr`, and `flush` characters are trapped locally and sent as TELNET protocol sequences to the remote side.

If LINEMODE has ever been enabled, then your `susp` and `eof` are also sent as TELNET protocol sequences, and `quit` is sent as a TELNET ABORT instead of as a BREAK. There are options (see `toggle autoflush` and `toggle autosynch` below) that cause this action to flush subsequent terminal output (until the remote host acknowledges the TELNET sequence) and to flush previous terminal input (in the case of `quit` and `intr`).

While you're connected to a remote host, you can enter `telnet`'s command mode by entering the `telnet` escape sequence (initially **Ctrl**–**]**). The normal terminal editing conventions are available in command mode.

Telnet commands:

In the following commands, you need to type only enough of each command to uniquely identify it; this is also true for arguments to the `mode`, `set`, `toggle`, `unset`, `environ`, and `display` commands.

`close`

Close a TELNET session and return to command mode.

`display` ***argument...***

Display all, or some, of the set and toggle values (see below).

`mode` ***type***

The *type* argument can have one of several values, depending on the state of the TELNET session. The remote host is asked for permission to go into the requested mode. If the remote host can enter that mode, the mode is entered.

`character`

Disable the TELNET LINEMODE option; if the remote side doesn't understand the LINEMODE option, enter character-at-a-time mode instead.

`line`

Enable the TELNET LINEMODE option; if the remote side doesn't understand the LINEMODE option, attempt to enter old-line-by-line mode instead.

Copyright © 2014, QNX Software Systems Limited

[-]isig

Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. LINEMODE must be enabled.

[-]edit

Attempt to enable (disable) the EDIT mode of the LINEMODE option. LINEMODE must be enabled.

[-]softtabs

Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. LINEMODE must be enabled.

[-]litecho

Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. LINEMODE must be enabled.

?

Print help information for the `mode` command.

open ***host*** **[[-l]** ***user*****]** **[-*****port*****]**

Open a connection to the named host. If no port number is specified, `telnet` tries to contact a TELNET server at the default port. The host may be specified either as a hostname (see the */etc/hosts* (p. 946) file) or as an Internet address specified in the "dot notation" (see the Internet address manipulation routines, *inet*()* in the QNX Neutrino *C Library Reference*).

With the -l option, you can specify the username to be passed to the remote system via the ENVIRON option.

When connecting to a nonstandard port, `telnet` omits any automatic initiation of TELNET options. When *port* is preceded by a minus sign, the initial option negotiation is done. Once `telnet` establishes a connection, it opens the `.telnetrc` file that's in your home directory.

In this file, lines beginning with a # are comments. Blank lines are ignored. Lines that begin without whitespace are the start of a machine entry. The first thing on the line is the name of the machine that's being connected to. The rest of the line and the successive lines that begin with whitespace are assumed to be `telnet` commands; they're processed as if you had typed them manually at the `telnet>` command prompt.

quit

Copyright © 2014, QNX Software Systems Limited

Close any open TELNET session and exit `telnet`. An end-of-file (in command mode) also closes a session and exits.

send ***arguments***

Send one or more special character sequences to the remote host. You can specify the following arguments (more than one argument may be specified at a time):

`abort`

Send the TELNET ABORT (Abort processes) sequence.

`ao`

Send the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all its output to the user's terminal.

`ayt`

Send the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

`brk`

Send the TELNET BRK (Break) sequence, which may have significance to the remote system.

`ec`

Send the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

`el`

Send the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

`eof`

Send the TELNET EOF (end-of-file) sequence.

`eor`

Send the TELNET EOR (end-of-record) sequence.

`escape`

Send the current `telnet` escape character (initially **Ctrl**–**]**).

`ga`

Copyright © 2014, QNX Software Systems Limited

Send the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

getstatus

If the remote side supports the TELNET STATUS command, send the subnegotiation to request that the server send its current option status.

ip

Send the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

nop

Send the TELNET NOP (No OPeration) sequence.

susp

Send the TELNET SUSP (SUSPend process) sequence.

synch

Send the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data — it might not work if the remote system is a 4.2BSD system. If it doesn't work, a lowercase "r" might be echoed on the terminal.

?

Print help information for the send command.

set ***argument value*****, unset** ***argument value***

The set command sets any one of a number of telnet variables to a specific value or to TRUE. The special value off turns off the function associated with the variable — it's equivalent to using the unset command. The unset command disables or sets to FALSE any of the specified functions.

You can query the values of variables with the display command.

The telnet variables that can be set or unset, *but not toggled*, are listed here. Note that you also explicitly set or unset any of the toggle command's variables.

ayt

Copyright © 2014, QNX Software Systems Limited

If `telnet` is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT sequence is sent to the remote host. The initial value for the *Are You There* character is the terminal's status character.

`echo`

In old-line-by-line mode, this value (initially **Ctrl**–**E**) toggles between doing local echoing of entered characters (for normal processing) and it (e.g. for entering a password).

`eof`

If `telnet` is operating in LINEMODE or old-line-by-line mode, this character is sent to the remote system if entered as the first character on a line. The initial value of the `eof` character is the terminal's EOF character.

`erase`

If `telnet` is in localchars mode (see `toggle localchars` below) and in character-at-a-time mode, then when this character is typed, a TELNET EC sequence (see `send ec` above) is sent to the remote system. The initial value for the erase character is the terminal's erase character.

`escape`

This is the `telnet` escape character; entering it switches you to `telnet`'s command mode when `telnet` is connected to a remote system. (Initially, the character is **Ctrl**–**]** to switch).

`flushoutput`

If `telnet` is in localchars mode (see `toggle localchars` below) and the `flushoutput` character is typed, a TELNET AO sequence (see `send ao` above) is sent to the remote host. The initial value for the flush character is the terminal's flush character.

`forw1` or `forw2`

If `telnet` is operating in LINEMODE, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's `eol` and `eol2` characters.

`interrupt`

Copyright © 2014, QNX Software Systems Limited

If `telnet` is in localchars mode (see `toggle localchars` below) and the interrupt character is typed, a TELNET IP sequence (see `send ip` above) is sent to the remote host. The initial value for the interrupt character is the terminal's intr character.

`kill`

If `telnet` is in localchars mode (see `toggle localchars` below) and in character-at-a-time mode, then when this character is typed, a TELNET EL sequence (see `send el` above) is sent to the remote system. The initial value for the kill character is the terminal's kill character.

`lnext`

If telnet is in LINEMODE or old-line-by-line mode, then this character is the terminal's lnext character. The initial value for the lnext character is the terminal's lnext character.

`quit`

If `telnet` is in localchars mode (see `toggle localchars` below) and the quit character is typed, a TELNET BRK sequence (see `send brk` above) is sent to the remote host. The initial value for the quit character is the terminal's quit character.

`reprint`

If `telnet` is in LINEMODE or old-line-by-line mode, then this character is the terminal's reprint character. The initial value for the reprint character is the terminal's reprint character.

`start`

If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is the terminal's start character. The initial value for the kill character is the terminal's start character.

`stop`

If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is the terminal's stop character. The initial value for the kill character is the terminal's stop character.

`susp`

If `telnet` is in localchars mode, or if LINEMODE is enabled, and the suspend character is typed, a TELNET SUSP sequence (see

Copyright © 2014, QNX Software Systems Limited

send susp above) is sent to the remote host. The initial value for the suspend character is the terminal's suspend character.

tracefile

This the file to which the output is written (when netdata or option tracing is TRUE). If it's set to -, tracing information is written to standard output (the default).

worderase

If telnet is operating in LINEMODE or old-line-by-line mode, then this character is the terminal's word-erase character. The initial value for the word-erase character is the terminal's word-erase character.

?

Display the legal set (unset) commands.

environ *argument...*

The environ command manipulates the variables that may be sent through the TELNET ENVIRON option. The initial set of variables is taken from the user's environment; only the ***DISPLAY*** and ***PRINTER*** environment variables are exported by default. Valid arguments are:

define *variable value*

Define this *variable* to have this *value*. Any variables defined by this command aren't automatically exported. To include tabs and spaces in *value*, enclose it in single or double quotes.

send *variable*

Send *variable* to the remote site.

undefine *variable*

Remove *variable* from the list of environment variables.

export *variable*

Mark *variable* to be exported to the remote side.

unexport *variable*

Mark *variable* **not** to be exported unless explicitly asked for by the remote side.

Copyright © 2014, QNX Software Systems Limited

list

List the current set of environment variables. Those marked with a * are sent automatically; other variables are sent only if explicitly requested.

?

Print help information for the `environ` command.

toggle *flag...*

Toggle between TRUE and FALSE various flags that control how `telnet` responds to events. You can explicitly set these flags to TRUE or FALSE with the `set` and `unset` commands listed above. To query the state of these flags, use the `display` command. Note that you can specify more than one flag.

Valid arguments are:

autoflush

If `autoflush` and `localchars` are both TRUE, and the `ao` or `quit` characters are recognized (and transformed into TELNET sequences; see `set` above for details), refuse to display any data on the user's terminal until the remote system acknowledges (via a TELNET TIMING MARK option) that it has processed those TELNET sequences. If the terminal user hasn't done an `stty noflsh`, the initial value for this toggle is TRUE; otherwise it's FALSE. See *stty* (p. 1863).

autosynch

If `autosynch` and `localchars` are both TRUE, and either the `intr` or `quit` character is typed (see `set` above), send the resulting TELNET sequence and follow it with the TELNET SYNCH sequence. This procedure should cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary

Enable or disable the TELNET BINARY option on *both* input and output.

inbinary

Copyright © 2014, QNX Software Systems Limited

Enable or disable the TELNET BINARY option on input only.

outbinary

Enable or disable the TELNET BINARY option on output only.

crlf

If this is TRUE, send carriage returns as CR LF. If this is FALSE, send carriage returns as CR NUL. The initial value for this toggle is FALSE.

crmod

Toggle carriage-return mode. When this mode is enabled, most carriage-return characters received from the remote host are mapped into a carriage return followed by a line feed. This mode doesn't affect characters you type, only those received from the remote host. This mode is useful only if the remote host sends only carriage returns and never sends line feeds. The initial value for this toggle is FALSE.

debug

Toggle socket-level debugging (useful only to the superuser). The initial value for this toggle is FALSE.

localchars

If this is TRUE, then the `flush`, `interrupt`, `quit`, `erase`, and `kill` characters (see `set` above) are recognized locally, and should be transformed into appropriate TELNET control sequences (respectively `ao`, `ip`, `brk`, `ec`, and `el`; see `send` above). The initial value for this toggle is TRUE in old-line-by-line mode, and FALSE in character-at-a-time mode. When the LINEMODE option is enabled, the value of `localchars` is ignored and assumed to be always TRUE. If LINEMODE has ever been enabled, then `quit` is sent as `abort`, and `eof` and `susp` are sent as `eof` and `susp` (see `send` above).

netdata

Toggle the display of all network data (in hex format). The initial value for this toggle is FALSE.

options

Copyright © 2014, QNX Software Systems Limited

Toggle the display of some internal `telnet` protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

`prettydump`

If the `netdata` and `prettydump` toggles are both enabled, the output from the `netdata` command is formatted in a more user-readable format. Spaces are put between each character in the output, and the beginning of any TELNET escape sequence is preceded by a * to aid in locating it.

`skiprc`

When the `skiprc` toggle is TRUE, `telnet` skips the reading of the `.telnetrc` file in the users home directory when connections are opened. The initial value for this toggle is FALSE.

`termdata`

Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is FALSE.

`?`

Display the legal toggle commands.

`z`

Suspend `telnet`. This command works only if you're using `csh`.

`!` [*command*]

Execute a single command in a subshell on the local system. If *command* is omitted, an interactive subshell is invoked.

`status`

Show the current status of `telnet`. This includes the peer you're connected to as well as the current mode.

`?` [*command*]

Get help. With no arguments, `telnet` prints a help summary. If a command is specified, `telnet` prints the help for that command.

Based on:

RFC 854

Files:

Copyright © 2014, QNX Software Systems Limited

$HOME/.telnetrc

User-customized startup values for `telnet`.

The `telnet` utility requires the `libsocket.so` shared library.

Environment variables:

The `telnet` utility uses at least the ***HOME***, ***SHELL***, ***DISPLAY***, and ***TERM*** environment variables. The environment variables may be propagated to the other side via the TELNET ENVIRON option.

License:

This utility is based on copyright software of the Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Caveats:

On some remote systems, you have to turn echo off manually when you're in old-line-by-line mode.

In old-line-by-line mode or LINEMODE, the terminal's EOF character is recognized (and sent to the remote system) only when it's the first character on a line.

Copyright © 2014, QNX Software Systems Limited

telnetd

DARPA TELNET protocol daemon (UNIX)

You must be `root` to start this daemon.

Syntax:

```
telnetd [-debug] [-D (options|report|netstat|ptydata)]
        [-h] [-n] [-U] [-46] [port]
```

Runs on:

QNX Neutrino

Options:

-4 or -6

The address family to use for -debug mode. During normal operation (called from `inetd`), `telnetd` will use the file descriptor passed from `inetd`.

-D *modifier*

Print debugging information. You can specify any of the following:

`options`

Print information about the negotiation of TELNET options.

`report`

Print the same information as for `options`, plus some additional information about what processing is going on.

`netdata`

Display the data stream received by `telnetd`.

`ptydata`

Display data written to the pty.

-debug

Normally, `telnetd` is started automatically through `inetd`; this option lets you start `telnetd` manually, blocking on *port*, waiting for a connection.

Copyright © 2014, QNX Software Systems Limited

-h

Disable the printing of host-specific information before logging in has been completed.

-n

Disable the keepalive option.

-U

Refuse connections from addresses that can't be mapped back into a symbolic name.

port

The port to use in -debug mode (must come last).

Description:

The `telnetd` daemon is a server that supports the DARPA-standard TELNET virtual-terminal protocol.

The `telnetd` daemon is started when *`inetd`* (p. 977) receives a service request to connect to the TELNET port (`inetd` listens for service requests specified in the *`inetd.conf`* (p. 979) file at a port defined in the *`services`* (p. 1744) file).

By specifying the -debug option, you can start up `telnetd` manually instead of through `inetd`. If you start `telnetd` this way, you can use the *port* argument to run `telnetd` on an alternate TCP port number.

You can use the -D option for debugging. This option lets `telnet` print debugging information to the connection, letting you see what `telnetd` is doing.

The `telnetd` daemon operates by allocating a pseudo-terminal device for a client, then creating a login process that has the slave side of the pseudo-terminal as standard input, standard output, and standard error. The `telnetd` daemon manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

If you get a message saying that all network ports are in use, you've either run out of pseudo devices or you haven't started *`devc-pty`* (p. 288). The `telnetd` daemon looks only at the pseudo-terminal devices named `/dev/pty[pqrs][0-f]`, no matter how many others are created.

When a TELNET session is started up, `telnetd` sends TELNET options to the client side that indicate a willingness to do remote echoing of characters, to suppress go-ahead, and to do remote flow control, as well as to receive terminal-type information, terminal-speed information, and window-size information from the remote client. If

Copyright © 2014, QNX Software Systems Limited

the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in *cooked* mode, and with XTABS and CRMOD enabled.

The `telnetd` daemon is willing to do:

- echo
- binary
- suppress go ahead
- timing mark

It's also willing to have the remote client do:

- line mode
- binary
- terminal type
- terminal speed
- window size
- toggle flow control
- environment
- X display location
- suppress go ahead

Name resolving issues

It is not mandatory for `telnetd` to have access to name-resolving capabilities. If it does have access to these capabilities, `telnetd` does a reverse name lookup (IP to hostname) of the telnet client.

If you decide to use a nameserver, make sure that the nameserver configuration is correct. If it isn't, there could be a delay of up to 1.5 minutes a login prompt is returned to the client, while the socket library's name resolver attempts to resolve the IP to a hostname.

Typical configuration for running `telnetd` on an embedded target

As mentioned before, in a host system environment, you run `telnetd` by just typing *`inetd`* (p. 977) in the command line. If you want to run `telnetd` on an embedded target, you need to copy the following files to your target:

- the appropriate variant of *`io-pkt*`* (p. 1007)
- Ethernet driver shared object (i.e. `devn-ne2000.so`)
- `libsocket.so`
- *`devc-pty`* (p. 288) for pseudo-tty support. You should use the -n option for this daemon to specify the number of sessions; each `telnet` session uses one pseudo device.

Copyright © 2014, QNX Software Systems Limited

- *ifconfig* (p. 957) and *route* (p. 1680) — both these utilities are used to configure your network interface. These utilities could be replaced by `dhcp.client`.
- *inetd* (p. 977) — this daemon is the Internet super-server.
- *inetd.conf* (p. 979) — TCP service daemons are listed in this file. Note that the descriptions in the default `inetd.conf` file are commented out; uncomment the ones that you want to use.

  The location of `inetd` and `inetd.conf` are `/usr/sbin`, and `/etc` respectively. If you decide to move `inetd.conf` to another location, you need to tell `inetd` in the command line.

The minimal `inetd.conf` contents to make `telnetd` work are as follows:

```
inetd.conf = {
# Internet services syntax:
# <service_name>  <socket_type> <proto> <flags> <user>  <server_pathname><args>

telnet stream tcp nowait root /usr/sbin/telnetd in.telnetd
telnet stream tcp6 nowait root /usr/sbin/telnetd telnetd
}
```

- `telnetd` — the location for this is usually `/usr/sbin/telnetd`. If you move it to another location, modify `inetd.conf` accordingly.
- */etc/services* (p. 1744) — maps service names such as telnet to a port number (23).
- `/bin/login`

  You can link `login` to `sh` or another binary if you don't want to actually log in, but you must include `/etc/passwd` if you do want to log in.

- `/etc/termcap`
- `/usr/lib/terminfo`
- `/bin/sh`
- */etc/hosts* (p. 946) or */etc/nsswitch.conf* (p. 1369) or `CS_RESOLVE`

  For more information, see "*Name resolving issues* (p. 1946)," above.

To configure the interface from a shell prompt, either use `ifconfig` and `route` or `dhcp.client`. You can then start `inetd`.

Based on:

RFC 854

Files:

The `telnetd` daemon requires the `libsocket.so` shared library.

Copyright © 2014, QNX Software Systems Limited

License:

This utility is based on copyright software of the WIDE project and of the Regents of the University of California; for licensing information, see the Third Party License Terms List at *http://licensing.qnx.com/third-party-terms/*.

Caveats:

Because of bugs in the original 4.2 BSD `telnet`, `telnetd` performs some protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD telnet.

Binary mode has no common interpretation except between similar operating systems (UNIX in this case).

The terminal-type name received from the remote client is converted to lowercase.

The `telnetd` daemon never sends TELNET go-ahead commands.

Copyright © 2014, QNX Software Systems Limited

termdef

Display or set the terminal type (QNX)

Syntax:

```
termdef [-c command] [-I] [-e] [terminal_type]
termdef -s [terminal_type]
termdef - [terminal_type]
```

Runs on:

QNX Neutrino

Options:

-

Write the current terminal type to the standard output. If a terminal type is specified, write the new type to the standard output.

-c *command*

After setting up the terminal type, start up *command* instead of invoking the */bin/login -p* (p. 1121) command.

-e

Instead of prompting for a terminal type, use the terminal type currently defined by the ***TERM*** environment variable. With this option, `termdef` leaves the ***TERM*** variable alone, but sends any necessary initialization sequences to the terminal and sets the appropriate *stty* (p. 1863) sequences for command-line editing.

-I

Don't send terminal initialization sequences. When -I isn't given, `termdef` writes any terminal initialization sequences to the standard output, if required for the terminal type.

-s

Write to the standard output the Shell command for setting ***TERM*** to the new terminal type. If no terminal type is specified, you'll be prompted for one. If you enclose the `termdef` command in backquotes, this option will let you set the ***TERM*** variable without exiting the Shell (see examples).

Copyright © 2014, QNX Software Systems Limited

terminal_type

The terminal type to be used. If *terminal_type* is included, the user isn't prompted for a terminal type. This overrides the -e option.

Description:

The `termdef` utility is used for setting and querying the terminal type. It can be used from *`tinit`* (p. 1964), where it will establish the terminal type and then *exec()* into another command (usually *`login`* (p. 1121)). It can also be used inside the shell to set or query the terminal type currently being used (`termdef -` or `termdef -s`).

Using the specified terminal type (or the type currently defined by the ***TERM*** environment variable if -e is given), `termdef`:

- sets the ***TERM*** variable
- sends any necessary terminal initialization sequences to the terminal
- sets the *`stty`* (p. 1863) sequences for command-line editing

Having configured the tty, `termdef` *exec()*s into a command that inherits the ***TERM*** environment variable from `termdef`. By default, this command is:

```
/bin/login -p
```

To use another command, you specify the -c option.

If a terminal type isn't specified or if -e isn't given, the user will be prompted for the terminal type. The user can obtain a list of valid terminal types by pressing **Enter** or by entering an invalid terminal type.

Specifying the -e option or a terminal type on the command line normally prevents the user from being prompted. However, if the terminal type defined by ***TERM*** is invalid, `termdef` displays a warning message and the user is prompted for a terminal type, as if -e weren't specified.

When invoked without the - or -s options, `termdef` establishes a terminal type and *exec()*s into another command (*`login`* (p. 1121), by default).

Using `termdef` from the shell

You can use `termdef` from the shell in a manner similar to the UNIX `tset` utility. To display the current terminal type, use:

```
termdef -
```

To change the terminal type, use one of the following:

This command:	Will:
`termdef` - *terminal_type*	Write the new *terminal_type* to the standard output

Copyright © 2014, QNX Software Systems Limited

This command:	Will:
`termdef` -s *terminal_type*	Write the shell command for setting ***TERM*** to the standard output
`termdef` -s	Do the same as the above command but prompt for the terminal type

Examples:

sysinit examples in conjunction with *tinit* (p. 1964)

Set up `ser2` and `ser3` to prompt for a terminal type, and *exec()* into *login* (p. 1121) once the terminal type is established:

```
tinit -c "termdef" -T ser2 ser3 &
```

Assuming `ser1` has a VT100 terminal on it (hardwired), have `termdef` set up the tty and ***TERM*** variable, then *exec()* into *login* (p. 1121), without prompting for terminal type:

```
tinit -c "termdef vt100" -T ser1 &
```

or:

```
tinit -c "termdef -e" -T ser1 TERM=vt100 &
```

Command-line examples

Display the terminal type:

```
termdef -
```

or:

```
echo $TERM
```

Change the terminal type to VT100:

```
export TERM=`termdef - vt100`
```

or:

```
`termdef -s vt100`
```

Prompt the user to specify a new terminal type:

```
`termdef -s`
```

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

The `termdef` command was started with valid command parameters, but may or may not have been successful. If unsuccessful, `termdef` writes diagnostic messages to standard error.

>0

Invalid command-line options were supplied.

Copyright © 2014, QNX Software Systems Limited

textto

Convert text files to QNX 2, QNX 4, UNIX, or DOS format (QNX Neutrino)

Syntax:

```
textto [-c|l|r] [-qz] file...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-c

Convert the specified files to DOS format (CR/LF).

-l

("el") Convert the specified files to QNX 4 and UNIX format (LF). By default, if none of the options are specified, -l is assumed.

-q

Be quiet; don't write the names of files being processed.

-r

Convert the specified files to QNX 2 format (RS).

-z

Strip all ^Z codes from the file.

file

The name of the file to be converted. You can use wildcards in the name.

Description:

The `textto` utility lets you convert text files to QNX 2, QNX 4/UNIX, or DOS format. You can use the -z option to strip ^Z codes from DOS-based files. Some DOS programs use this character as an end-of-file indicator.

Examples:

Change all QNX 2 record separators in `prog.c` to QNX 4 line feeds:

```
textto -l prog.c
```

Copyright © 2014, QNX Software Systems Limited

Change all QNX 4 line feeds in all of the `.h` files in the current directory to QNX 2 record separators:

```
textto -r *.h
```

Change all QNX 4 line feeds in all of the `.txt` files in the current directory to DOS CR/LFs:

```
textto -c *.txt
```


Copyright © 2014, QNX Software Systems Limited

tftp

Trivial file transfer program

Syntax:

```
tftp [-e] [host] [port]
```

Runs on:

QNX Neutrino

Options:

-e

Use binary transfer mode and set the extended options, as if `tout`, `tsize`, and `blksize 65464` had been given.

host

The host to connect to.

port

The port to connect to.

Description:

The `tftp` utility is the user interface to the Internet TFTP (*T*rivial *F*ile *T*ransfer *P*rotocol), which lets you transfer files to and from a remote machine. If you specify a remote host (and an optional port) on the command line, `tftp` uses that host (and port) as the default for future transfers (see the `connect` command below). The client host must have an entry in the */etc/services* (p. 1744) file to provide `tftp` service.

Remote filenames must start with a `/`. If a list of directories is given to `tftpd`, filenames must be in that list. To `get` a file, the file must have world-read privileges; to `put` a file, the file must have world-write privileges on the remote machine.

Commands:

Once `tftp` is running, it issues a prompt and accepts the following commands:

? command_name...

Print help information.

Copyright © 2014, QNX Software Systems Limited

ascii

Shorthand for "`mode ascii`" (see `mode` below).

binary

Shorthand for "`mode binary`" (see `mode` below).

blksize *blk-size*

Set the blocksize to *blk-size* octets (8-bit bytes). Since the number of blocks in a `tftp get` or `put` is 65535, the default block size of 512 bytes allows only a maximum of just under 32 MB to be transferred. The value given for *blk-size* must be between 8 and 65464, inclusive. Note that many servers don't respect this option.

connect *host* **[***port***]**

Set the *host* (and optionally *port*) for transfers. Note that the TFTP protocol, unlike FTP, doesn't maintain connections between transfers. That is, the `connect` command doesn't actually create a connection, but merely remembers what host to use for transfers. You don't have to use the `connect` command; you can specify the remote host as part of the `get` or `put` commands.

get *file* **or get** *remotefile localfile* **or get** *file1 file2 ... fileN*

Get a file or set of files from the specified sources. The source can be in one of two forms: a filename on the remote host (if the host has already been specified), or a string of the form *hosts* **:** *filename* to specify both a host and file at the same time. If you use the latter form, the last hostname specified becomes the default.

mode *transfer_mode*

Set the mode for transfers to either `ascii` or `binary` (default is `ascii`).

put *file* **or put** *localfile remotefile* **or put** *file1 file2 ... fileN remote_directory*

Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host (if the host has already been specified), or a string of the form *hosts* **:** *filename* to specify both a host and file. If you use the latter form, the hostname specified becomes the default; if you use the *remote_directory* form, the remote host is assumed to be a Unix box.

If you need to specify IPv6 numeric addresses for the hosts, enclose them in square brackets (**[***hosts***]:***filename*) to disambiguate the colon.

Copyright © 2014, QNX Software Systems Limited

quit

Exit `tftp`. An end-of-file also exits.

rexmt *retransmission_timeout*

Set the per-packet retransmission timeout (specified in seconds).

status

Show the current status.

timeout *total_transmission_timeout*

Set the total transmission timeout (specified in seconds).

tout

Toggle the `timeout` option. If you use this option, the client passes its retransmission-timeout to the server. Note that many servers don't respect this option.

trace

Toggle packet tracing.

tsize

Toggle the `tsize` option. If you enable this option, the client passes and requests the size of a file at the beginning of a file transfer. Note that many servers don't respect this option.

verbose

Toggle verbose mode.

Files:

The `tftp` utility requires the `libsocket.so` shared library.

Caveats:

Since there's no user-login or validation within the TFTP protocol, the remote site should have some file-access restrictions — the exact methods for implementing these vary.

Copyright © 2014, QNX Software Systems Limited

tftpd

DARPA trivial file transfer protocol daemon

You must be `root` to start this daemon.

Syntax:

```
tftpd [-dln] [-g group] [-p port] [-s directory]
      [-u user] [directory ...]
```

Runs on:

QNX Neutrino

Options:

-d

Increase the debugging level.

-g *group*

Change the group ID to that of *group* on startup. If you don't specify this option, the group ID is set to that of the user specified with the -u option.

-l

Log all requests using *syslogd* (p. 1885).

-n

Suppress negative acknowledgement of requests for nonexistent relative filenames.

-p *port*

The UDP port to bind when running `tftpd` standalone.

-s *directory*

The root directory that you want `tftpd` to change to on startup. This is recommended for security reasons (so that files other than those in the `/tftpboot` directory aren't accessable). If the remote host passes the directory name as part of the file name to transfer, you may have to create a symbolic link from `tftpboot` to `.` under `/tftpboot`.

Copyright © 2014, QNX Software Systems Limited

-u *user*

Change the user ID to that of *user* on startup. If you don't specify this option, the user defaults to `nobody`. If you don't specify -g, `tftpd` uses the group ID of *user* as well.

directory

The name of the directory where files are accessed.

Description:

The `tftpd` daemon is a server that supports the DARPA Trivial File Transfer Protocol.

The `tftpd` daemon is started when *`inetd`* (p. 977) receives a service request at the port indicated by the *tftp* entry (`inetd` listens for service requests specified in the *`inetd.conf`* (p. 979) at a port defined in the *`services`* (p. 1744) file). Note that the descriptions in the default `inetd.conf` file are commented out; uncomment the ones that you want to use.

Because `tftp` doesn't require an account or password on the remote system, `tftpd` allows only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable.

This extends the concept of "public" to include all users on all hosts that can be reached through the network—this may *not* be appropriate on all systems, and its implications should be considered before enabling the `tftp` service. The daemon should have the user ID with the lowest possible privilege.

Filenames must start with a `/`. If a list of directories is given to `tftpd`, filenames must be in that list. To `get` a file, the file must have world-read privileges; to `put` a file, the file must have world-write privileges.

The *`bootpd`* (p. 65) utility works with `tftpd` to provide the client with a boot image.

Based on:

RFC 1350

Files:

The `tftpd` daemon requires the `libsocket.so` shared library.

Copyright © 2014, QNX Software Systems Limited

tic

Terminfo compiler (UNIX)

Syntax:

```
tic [-c] [-i] [-v n] file
```

Runs on:

QNX Neutrino

Options:

-c

Perform only a syntax check of the input file. No output file is produced.

-i

Ignore errors when compiling terminfo files from other systems that describe capabilities not currently supported in `/usr/lib/terminfo`.

-v *n*

Set verbose output level. Progress information is output to *stderr* as `tic` works though the source file. The integer argument controls the verbosity of the report, 1 providing the least detail and 10 providing the most. The default is 0, providing "quiet" compilation.

file

The pathname of a file that contains one or more terminal descriptions in `terminfo` source format. Each description in the file describes a particular terminal. When a use=*entry_name* field is encountered within a terminal description, `tic` reads the previously compiled description from the `/usr/lib/terminfo` directory to finish the entry. This allows terminal descriptions that are only slightly different from one previously defined to be expressed as an extension to that previous description. If the environment variable ***TERMINFO*** is set, the directory named is used instead of the `/usr/lib/terminfo` directory.

Copyright © 2014, QNX Software Systems Limited

Description:

The `tic` utility translates a terminfo file from the source format into the compiled format, ready to use by applications running on the terminal type named in the terminfo source file.

The compiled `terminfo` capability files are stored in single-character directories, under the `/usr/lib/terminfo` directory. For example, the VT100 terminfo file is stored in `/usr/lib/terminfo/v/vt100`.

You can use the *`infocmp`* (p. 987) utility to convert a binary, compiled terminfo file back into a source format that you can modify, and then recompile it with `tic`.

Some error messages produced by `tic` indicate the line of the source file in error and the terminal name being processed at that point.

For more information, see Strang, John, Linda Mui, and Tim O'Reilly. 1988. *termcap & terminfo*. Sebastopol, CA: O'Reilly and Associates. ISBN 0937175226.

Environment variables:

`TERMINFO`

When defined, the compiled output from `tic` is placed under the directory named instead of in the `/usr/lib/terminfo` directory.

Exit status:

0

The input file was compiled successfully.

>0

An error occurred.

Caveats:

The total size of a compiled terminfo description cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

Terminal names longer than 14 characters are truncated to 14 characters.

Copyright © 2014, QNX Software Systems Limited

time

Determine the execution time of a command (UNIX)

Syntax:

`time` *command*

Runs on:

QNX Neutrino

Options:

command

The command, including any of its options, for which the execution time is to be determined.

Description:

The `time` utility and the `time` shell built-in determine the execution time of *command*, which must be specified. When execution of *command* is complete, the `time` utility displays the total elapsed time for execution, the time spent in the system, and the time spent in executing *command*. Times are reported in hours, minutes, and seconds.

Times aren't always accurate to the milliseconds as displayed, depending on the tick size.

Examples:

Time the execution of `dcheck -b2048 /dev/hd0`:

```
time dcheck -b2048 /dev/hd0
```

Exit status:

The `time` utility exits with the exit status of the *command* that was timed.

Caveats:

The `time` command doesn't obtain execution information from a remote node. To time remote programs, use the *on* (p. 1417) command to run `time` on the same machine as the program being timed, e.g.:

```
on -n 61 time sleep 30
```


Copyright © 2014, QNX Software Systems Limited

NOT:

```
time on -n 61 sleep 30
```

Copyright © 2014, QNX Software Systems Limited

tinit

Terminal initialization

Syntax:

```
tinit [-f file] [-pt] [env_var=value...]
```

Runs on:

QNX Neutrino

Options:

-f file

The configuration file that defines the commands to start (see below). The default is `/etc/config/ttys`.

-t

Don't mask the suspend signal (`SIGTSTP`) in spawned process.

env_var=value

Set the environment variable *env_var* to *value* and add it to the environment.

Description:

The `tinit` utility lets you bring up *login* (p. 1121) (or other programs) on devices. You normally use `tinit` to invoke `login` on the console(s) and serial terminals.

The `tinit` utility runs as a background process, and is nearly always started in the `/etc/rc.d/rc.sysinit` file.

After creating the specified programs as its children, `tinit` waits for any of them to terminate. When one of these commands terminates, `tinit` re-invokes the command on that device.

For example, let's assume that `tinit` started a `login` on `/dev/con1`. After logging in, the user is presented with a shell. Then, after executing any number of commands, the user decides to terminate the shell. At this point, `tinit` detects the termination and starts a new `login` on `/dev/con1`.

`ttys` configuration file

The `tinit` utility refers to a configuration file (`/etc/config/ttys` or the file you specified with the -f option) to obtain information about your terminal devices and the

Copyright © 2014, QNX Software Systems Limited

commands you want to start on them. This file contains lines of text, each line containing four fields that give the configuration information for one device, like this:

```
con1 "/bin/login" qansi-m on
```

Here's what the fields in this example define:

con1

The first field is the device name. Unless you define a complete pathname, starting with / (slash), `tinit` adds the prefix `/dev/` to the name. This example defines the device name as `/dev/con1`. If you wanted to specify a terminal connected to a serial port, this field might read `ser1`.

"/bin/login".

The second field contains a string that specifies the pathname of the command to be started on the device. This command can be anything you like; most often it will be either `login`, as in this default example.

qansi-m

The third field describes the terminal type. You can find a list of the terminal types you can specify in `/usr/lib/terminfo`.

on

The fourth and last field is currently ignored by `tinit`. It is there for possible future expansion.

The parsing of this file is fairly simple:

- Fields may be quoted or unquoted; quoted fields are enclosed by double quotation marks (").
- An unquoted field is separated from the next field by one or more spaces; a quoted field is separated from the next by its closing quote, followed by zero or more spaces.
- There's no escape character, so you can't include double quotation marks in a quoted field. You can use single quotation marks (') inside quoted fields.

The `tinit` utility is the root of all logins, so any environment variables it sets will be inherited; this makes the `tinit` command line a convenient place for you to set the environment variables you need, using the *env_var* options.

Files:

/etc/config/ttys

The terminal configuration file used by `tinit`.

Copyright © 2014, QNX Software Systems Limited

top

Display system usage (Unix)

Syntax:

```
top [-i number] [-d] [-n node] [-p priority]
```

Runs on:

QNX Neutrino

Options:

-d

Tailor the output for a dumb terminal. By default, `top` refreshes its output in place for each iteration. If you specify -d, `top` displays the output for each iteration after the output from the previous one.

-n *node*

Run `top` on the specified remote node.

-p *priority*

Run at the specified priority.

-i *number*

Run for the specified number of iterations. By default, `top` runs until you terminate it.

Description:

The `top` utility displays the system usage. Its output looks like this:

```
42 processes; 119 threads;
CPU states: 67.3% idle, 29.9% user, 2.7% kernel
Memory: 0 total, 368M avail, page size 4K

      PID   TID PRI STATE    HH:MM:SS    CPU  COMMAND
   593962     1  10 Rcv       0:00:04  16.45% firefox-bin
   278558     3  12 Rply      0:00:50   6.00% io-graphics
   278558     2  10 Rcv       0:00:00   2.15% io-graphics
        1    19  10 Run       0:00:03   1.52% kernel
        1    13  10 Rcv       0:00:00   1.18% kernel
     8200    11  10 Rcv       0:00:00   0.73% devb-eide
   114707     2  12 Rcv       0:00:02   0.71% io-display
   131092     2  21 Rcv       0:04:39   0.35% io-pkt-v4-hc

             Min        Max       Average
CPU idle:     45%        98%         67%
Mem Avail:   368MB      398MB       382MB
```


Copyright © 2014, QNX Software Systems Limited

```
Processes:    39         42         41
Threads:     104        119        111
```

Copyright © 2014, QNX Software Systems Limited

touch

Change file access and modification times (POSIX)

Syntax:

```
touch [-acm] [-r ref_file|-t time] file...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Change the access time of *file* to *time*, or to the current time if *time* isn't specified. Don't change the modification time unless the -m option is also specified.

-c

If *file* doesn't already exist, don't create *file* and don't write any diagnostic messages concerning this condition.

-m

Change the modification time of *file* to *time*, or to the current time if *time* isn't specified. Don't change the access time unless -a is also specified.

-r *ref_file*

Use the specified reference file's modification time instead of the current time.

-t *time*

Use the specified *time* instead of the current time, where *time* is a decimal number of the form:

[[*CC*]*YY*]*MMDDhhmm*[.*SS*]

The two-digit pairs in *time* represent the following:

CC

The first two digits of the year (i.e. the century).

YY

Copyright © 2014, QNX Software Systems Limited

The second two digits of the year.

MM

The month of the year (01-12).

DD

The day of the month (01-31).

hh

The hour of the day (00-23).

mm

The minute of the hour (00-59).

SS

The second of the minute (00-61).

Both *CC* and *YY* are optional. If you specify neither of these, the current year is assumed. If you specify *YY*, but you don't specify *CC*, *CC* is derived as follows:

If *YY* is:	*CC* becomes:
69-99	19
00-68	20

The resulting time is affected by the value of the ***TZ*** environment variable (see below). If the resulting time precedes 0 hours, 0 minutes, 0 seconds, January 1, 1970 Coordinated Universal Time (i.e. the Epoch), `touch` exits immediately with an error status.

The range for *SS* is 00-61 rather than 00-59 to allow for leap seconds. If *SS* is 60 or 61, and the resulting time — as affected by the ***TZ*** environment variable — doesn't refer to a leap second, the resulting time is one second after a time where *SS* is 59. If you don't give a value to *SS*, it's assumed to be 0.

file

The pathname of a file whose times are to be modified.

Copyright © 2014, QNX Software Systems Limited

Description:

The `touch` utility lets you change either the modification times or access times of files, or both.

If any file you specify doesn't exist, the file is created unless you specify the -c option. If no time is specified, the current time is used. The -a option changes only the access time of the file; the -m option changes only the modification time of the file.

If you don't specify any options, `touch` behaves as if you used the -a and -m options.

Examples:

Set `file1`'s access and modification time to the current system time:

```
touch file1
```

Change `file2`'s access and modification time to be the same as `file1`'s modification time:

```
touch -r file1 file2
```

Set `file3`'s access and modification time to December 25, 12:34, using the current year:

```
touch -t 12251234 file3
```

Environment variables:

TZ

Specifies the local time for the specified time zone.

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

tr

Translate characters (POSIX)

Syntax:

```
tr [-cs] [-r filename] string1 string2
tr [-cs] [-r filename] string1
tr -d [-c] [-r filename] string1
tr -ds [-c] [-r filename] string1 string2
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-c

Complement the set of characters in *string1* with respect to the universe of characters from `00` through `FF` hex. Characters in *string1* are copied unchanged while all other characters are translated.

-d

Delete all input characters in *string1* (or not in *string1* for -dc).

-r *filename*

(QNX Neutrino extension) Translate the named file in place (don't use *stdin*/*stdout*).

-s

Squeeze all output strings of one or more instances of a single character in *string1* to a single instance of the corresponding character in *string2*.

If you don't specify *string2*, `tr` squeezes instances of the characters in *string1* to a single instance of that character.

string1

Translation character string (translate from).

string2

Translation character string (translate to).

Copyright © 2014, QNX Software Systems Limited

If you specify both -d and -s, `tr` deletes instances of the characters in *string1* and squeezes instances of the characters in *string2* (i.e. `tr` doesn't translate in this case).

Description:

The `tr` utility copies the standard input to the standard output with substitution or deletion of selected characters. The options specified and the *string1* and *string2* operands control translations that occur while copying characters.

The default behavior is to replace each input character found in *string1* with the character at the same position in *string2*, while copying characters not in *string1* unchanged.

When *string2* is shorter than *string1*, *string2* is extended to the length of *string1* by duplicating the last character of *string2*. If *string2* is explicitly a string of zero length, it's padded with NUL characters.

The *string1* and *string2* operands often require quoting to avoid interpretation by the shell. Single quotes are usually the proper quoting mechanism.

Conventions for *string1* and *string2*

Use the following conventions in *string1* or *string2* or both to specify characters, character ranges, character classes, or collating elements:

character

Represents that character.

\octal

A backslash followed by 1, 2, or 3 octal digits represents a character with that encoded value.

\character

A backslash followed by any character except an octal digit represents that character.

c-c

Represents the range of characters between the range endpoints, inclusive.

[*c-c*]

(QNX Neutrino extension) The System V method of representing a range of characters.

Copyright © 2014, QNX Software Systems Limited

[:*class*:]

Represents all characters belonging to the defined character class. Allowable names for *class* are:

`alnum`, `alpha`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, `xdigit`

[.*cs*.]

Represents a collating symbol. Multicharacter collating symbols must be represented as collating symbols to distinguish them from a string of the same characters. This implementation allows an arbitrary string to be treated as a collating symbol (QNX Neutrino extension).

[*x*n*]**

Represents *n* repeated occurrences of the character or collating symbol *x*. This expression is valid only in *string2*. If *n* is omitted or is zero, it's interpreted as large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence. If *n* has a leading zero, it's interpreted as an octal value. Otherwise, it's interpreted as a decimal value.

Examples:

Convert all lowercase characters in the input to the corresponding uppercase characters:

```
tr '[:lower:]' '[:upper:]' <file1 >file2
```

Or

```
tr '[a-z]' '[A-Z]' <file1 >file2
```

Create a list of all words in `file1` one per line in `file2` where a word is taken to be a maximal string of letters (octal 012 is the code for newline):

```
tr -cs '[:alpha:]' '[\012*]' <file 1 >file2
```

Convert a DOS file into a UNIX file:

```
tr -d '\15' <infile >outfile
```

Exit status:

0

Success

1

An error occurred.

Copyright © 2014, QNX Software Systems Limited

tracelogger

Log tracing information into an event file

- You must be `root` to run this utility.
- In order to log trace events, your system must be running an instrumented version of *procnto* (p. 1586) (i.e. `procnto*-instr`).

Syntax:

```
tracelogger [-cEPRrw] [-A attribute] [-b num] [-d mode]
            [-F num] [-f file] [-k num] [-M -S size]
            [-n num] [-p addr] [-s num] [-v[v...]]
```

Runs on:

QNX Neutrino

Targets:

ARMv7, x86

Options:

-A attribute

Specify an attribute to add to the log. The *attribute* is in the form *name=value*, where the *name* and *value* are arbitrary strings.

You can't use the following in the name or value:

- `::`
- TRACE_
- =

If you do, `tracelogger` prints a message and ignores the attribute.

You can use one or more occurrences of this option to add information to identify the scenario that you're logging. For example:

```
tracelogger -s1 -A MACHINE_NAME=tx86 -A PERIOD=1s
```

-b num

Copyright © 2014, QNX Software Systems Limited

The maximum number of dynamic buffers to allocate in `tracelogger`; the default is 64. Each buffer has a size of approximately 11 KB.

-c

Operate in continuous mode; the default is to run in iterations mode.

-d *mode*

Currently, the *mode* is:

1 ("One")

Launch in daemon mode. Unless you also specify the -E option, `tracelogger` ignores all its other options, leaving your application to configure, start, and stop the tracing through calls to *TraceEvent()*.

-E

Extend the daemon (-d1) mode by honoring the other options to `tracelogger`. Your application still needs to call *TraceEvent()* to start capturing trace events.

-f *file*

The name of the file to store logged events in. The default is `/dev/shmem/tracebuffer.kev`.

-F *num*

Filtering for different *num* values, as follows:

0

Don't set any filtering.

1

Disable Kernel calls class.

2

Disable Interrupt class.

3

Disable Process class.

4

Disable Thread class.

Copyright © 2014, QNX Software Systems Limited

5

Disable VThread class.

6

Disable Communication class.

7

Disable System class.

You can specify more than one filter by using multiple -F options. For example, to disable both the Interrupt and VThread classes, specify -F2 -F5. For more information about these classes, see the Events and the Kernel chapter of the System Analysis Toolkit *User's Guide*.

-k ***num***

The number of buffers to allocate in the kernel; the default is 32. Each buffer has a size of approximately 16 KB.

-M

Map the log file directly instead of writing. If you use this option, the log file must be in shared memory, and you must use the -S option to specify the maximum file size.

-n ***num***

The number of iterations to log in iterations mode. The default is 32; specify 0 for unlimited iterations.

-P

Preserve the kernel instrumentation buffers in a shared memory block in `/dev/shmem`. If you specify this option, then when `tracelogger` exits, it doesn't deallocate the in-kernel buffer memory; you can then specify the -R option on subsequent runs to make `tracelogger` reuse the same buffers without reallocating memory.

-p ***addr***

Specify the address of the physical memory to use for kernel buffers.

-R

Reuse the kernel instrumentation buffers that were previously created when you used the -P option. If there are no buffers to reuse, `tracelogger` allocates the kernel buffers as usual.

Copyright © 2014, QNX Software Systems Limited

-r

Set the kernel buffer to ring mode. The default is linear mode.

-s *num*

The number of seconds to log for. The default is 0, specifying an unlimited duration.

-S *size*

The maximum size of the log file. Use `M` for megabytes or `K` for kilobytes. If you don't use `M` or `K`, the units are assumed to be bytes. You must specify this option if you use the -M option.

-v[v...]

Be verbose; more `v` characters cause more verbosity.

-w

Log wide events; the default is to log fast events.

Description:

The `tracelogger` daemon receives events from the instrumented kernel (*`procnto*-instr`* (p. 1586)) and saves them in a file or on a device for later analysis. By default, `tracelogger` saves the events in `/dev/shmem/tracebuffer.kev`. The `.kev` extension is short for "kernel events"; the Integrated Development Environment opens files with this extension in the System Profiler.

You can run `tracelogger` in the following modes:

-n *iterations*

The kernel logs events; `tracelogger` captures *iterations* buffers worth of data, and then terminates. This is the default, and *iterations* defaults to 32.

-r

Ring mode: the kernel logs events, but `tracelogger` doesn't capture them until it gets a `SIGINT` signal, or an application calls *TraceEvent()* with a command of `_NTO_TRACE_STOP`.

-d1

Daemon mode: the kernel doesn't log events, and `tracelogger` doesn't capture them, until an application calls *TraceEvent()* with a command of `_NTO_TRACE_START`. Logging continues until an application calls *TraceEvent()* with a command of `_NTO_TRACE_STOP`, or you terminate `tracelogger`.

Copyright © 2014, QNX Software Systems Limited

-s *seconds*

The kernel logs events; `tracelogger` captures them over the specified time.

-c

The kernel logs events; `tracelogger` captures them, and continues to do so until terminated.

The simplest way to configure the tracing is to start `tracelogger` in normal or extended daemon (-d1 -E) mode, using the command-line options to specify what to trace. In this case, `tracelogger` performs all initialization and runtime control of the instrumentation module in addition to its normal task of saving the trace buffers to the filesystem. If you use extended daemon mode, your application needs to call *TraceEvent()* to start capturing trace events.

For finer (and more flexible) control of the instrumentation, run `tracelogger` in daemon (-d1) mode, and call *TraceEvent()* from your application to specify what to trace, and when to start and stop tracing. Except for specifying the number of buffers, `tracelogger` doesn't perform any initialization of the instrumentation module, and its almost exclusive task is to log the "received" trace events to the filesystem.

On a multicore system, if the clocks on all processors aren't synchronized, then `tracelogger` will produce data with inconsistent timestamps, and the IDE won't be able to load the trace file. The IDE attempts to properly order the events in the trace file, and this can go awry if the timestamp data is incorrect.

The *`traceprinter`* (p. 1980) utility doesn't have any issues with such traces because it doesn't attempt to reorder the data and interpret it; it simply dumps the contents of each event.

Examples:

Start `tracelogger` in wide mode and display operational information on the console screen; store the logged data in the named file and stop logging when 12 trace buffers are full:

```
tracelogger -f /dev/shmem/my_tracebuffer.kev -n 12 -w
```

Start `tracelogger` in *ring* mode (background) using 5 internal buffers and wait for an asynchronous signal (e.g. **Ctrl**–**C**) to stop it:

```
tracelogger -r -b 5
```


Copyright © 2014, QNX Software Systems Limited

Run `tracelogger` in continous mode for 20 seconds:

```
tracelogger -c -s20
```

Exit status:

-1

An error occurred.

Errors:

Severe errors cause `tracelogger` to terminate.

Caveats:

Run only one instance of `tracelogger` at a time. Similarly, don't run `tracelogger` and use *qconn* (p. 1621) (under control of the IDE) to trace events at the same time.

Copyright © 2014, QNX Software Systems Limited

traceprinter

Display the contents of the instrumented kernel trace file

Syntax:

```
traceprinter [-nv] [-f file] [-o out_file]
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-f *file*

The filename of the file that stores the trace information to be displayed. The default is `/dev/shmem/tracebuffer`.

-n

Remove newline characters from printed argument lines.

-o *out_file*

The name of the file in which to store the data. By default, `traceprinter` sends its output to *stdout*.

-v

Display detailed information.

Description:

The `traceprinter` utility displays the contents of a trace file generated by *`tracelogger`* (p. 1974). The utility parses the linearly stored time events stored in the named trace file and sends the resulting formatted stream to standard output (or to the file identified by the -o option).

The formatted stream looks like this:

```
t:clk_time CPU:cpu [class: event: [P1: [P2: [P3: ... [Pn:]]]]]
```

The stream always shows the *clk_time* and *cpu* variables; the variables in brackets are optional.

clk_time

Copyright © 2014, QNX Software Systems Limited

The time offset in CPU clock cycles when the trace event was registered. The 64-bit variable is broken into two 32-bit hexadecimal numbers (*msb* and *lsb*). Apart from at the start of the trace and when the *lsb* rolls over, only the *lsb* number is shown.

cpu

A 2-digit decimal number representing the CPU that registered the event. The variable is always 00 unless you have more than one CPU in your system: if you have four CPUs, the CPU numbers range from 00 to 03. The CPU numbers are assigned to particular CPUs during initialization, when the startup programs probe the system (see *procnto** (p. 1586)).

Optional variables

The other variables are optional, depending on the tracing information logged. The *class* and *event* variables are followed by parameter variables, the number and type of parameters depend on the associated *class*/*event* pair and whether `tracelogger` used the fast or wide emitting mode. Each trace line contains one *class*, one or no *event*, and one or more parameter variables. The table below explains which parameters are shown for each combination of *class* and *event*.

Class	Event	Parameters
CONTROL	TIME	• P1: msb • P2: lsb (offset)
INT_ENTR, INT_EXIT	Interrupt number in hexadecimal (and decimal) notation	• P1: inkernel
THREAD	• THDEAD • THRUNNING • THREADY • THSTOPPED • THSEND • THRECEIVE • THREPLY • THSTACK • THWAITTHREAD • THWAITPAGE • THSIGSUSPEND	• P1: PID • P2: thread ID

Copyright © 2014, QNX Software Systems Limited

Class	Event	Parameters
	• THSIGWAITINFO • THNANOSLEEP • THMUTEX • THCONDVAR • THJOIN • THINTR • THSEM • THWAITCTXN • THNET_SEND • THNET_REPLY • THCREATE • THDESTROY	
VTHREAD	• VTHDEAD • VTHRUNNING • VTHREADY • VTHSTOPPED • VTHSEND • VTHRECEIVE • VTHREPLY • VTHSTACK • VTHWAITVTHREAD • VTHWAITPAGE • VTHSIGSUSPEND • VTHSIGWAITINFO • VTHNANOSLEEP • VTHMUTEX • VTHCONDVAR • VTHJOIN • VTHINTR • VTHSEM • VTHWAITCTXN • VTHNET_SEND • VTHNET_REPLY • VTHCREATE • VTHDESTROY	• P1: PID • P2: vthread ID

Copyright © 2014, QNX Software Systems Limited

Class	Event	Parameters
PROCESS	• PROCCREATE • PROCCREATE_NAME • PROCDESTROY • PROCDESTROY_NAME	• P1: parent PID • P2: PID • P3: process name (optional)
KER_CALL	Kernel call name	• P1: kernel call number • P2, P3, ... Pn: kernel call arguments
KER_EXIT	Kernel call name	• P1: kernel call number • P2, P3, ... Pn: kernel call return values

The number of arguments and return values depends on the event and whether the trace file was produced using fast or wide tracing mode. For more information, see the Current Trace Events and Data appendix of the System Analysis Toolkit *User's Guide*.

Examples:

Here's an example of the first few lines of output from `traceprinter`:

```
TRACEPRINTER version 0.94
 -- HEADER FILE INFORMATION --
       TRACE_FILE_NAME:: /scratch/quadlog
            TRACE_DATE:: Fri Jun  8 13:14:40 2001
       TRACE_VER_MAJOR:: 0
       TRACE_VER_MINOR:: 96
   TRACE_LITTLE_ENDIAN:: TRUE
         TRACE_ENCODING:: 16 byte events
       TRACE_BOOT_DATE:: Fri Jun  8 04:31:05 2001
  TRACE_CYCLES_PER_SEC:: 400181900
          TRACE_CPU_NUM:: 4
          TRACE_SYSNAME:: QNX
         TRACE_NODENAME:: x86quad.gp.qa
     TRACE_SYS_RELEASE:: 6.1.0
     TRACE_SYS_VERSION:: 2001/06/04-14:07:56
         TRACE_MACHINE:: x86pc
     TRACE_SYSPAGE_LEN:: 2440
 -- KERNEL EVENTS --
t:0x1310da15 CPU:01 CONTROL :TIME msb:0x0000000f, lsb(offset):0x1310d81c
t:0x1310e89d CPU:01 PROCESS :PROCCREATE_NAME
                      ppid:0
                       pid:1
                      name:./procnto-smp-instr
t:0x1310eee4 CPU:00 THREAD  :THCREATE      pid:1 tid:1
t:0x1310f052 CPU:00 THREAD  :THRUNNING     pid:1 tid:1
t:0x1310f144 CPU:01 THREAD  :THCREATE      pid:1 tid:2
t:0x1310f201 CPU:01 THREAD  :THREADY       pid:1 tid:2
t:0x1310f32f CPU:02 THREAD  :THCREATE      pid:1 tid:3
t:0x1310f3ec CPU:02 THREAD  :THREADY       pid:1 tid:3
t:0x1310f52d CPU:03 THREAD  :THCREATE      pid:1 tid:4
t:0x1310f5ea CPU:03 THREAD  :THRUNNING     pid:1 tid:4
t:0x1310f731 CPU:02 THREAD  :THCREATE      pid:1 tid:5
t:0x1310f7ee CPU:02 THREAD  :THRECEIVE     pid:1 tid:5
t:0x1310f921 CPU:03 THREAD  :THCREATE      pid:1 tid:6
```

Copyright © 2014, QNX Software Systems Limited

```
t:0x1310f9de CPU:03 THREAD  :THRECEIVE     pid:1 tid:6
t:0x1310fb0b CPU:01 THREAD  :THCREATE      pid:1 tid:7
t:0x1310fbc8 CPU:01 THREAD  :THRECEIVE     pid:1 tid:7
t:0x1310fd1d CPU:02 THREAD  :THCREATE      pid:1 tid:8
t:0x1310fdda CPU:02 THREAD  :THRECEIVE     pid:1 tid:8
t:0x1310ff35 CPU:02 THREAD  :THCREATE      pid:1 tid:9
t:0x1310fff2 CPU:02 THREAD  :THRECEIVE     pid:1 tid:9
t:0x131100cc CPU:01 THREAD  :THCREATE      pid:1 tid:10
t:0x13110189 CPU:01 THREAD  :THRECEIVE     pid:1 tid:10
t:0x131102d5 CPU:03 THREAD  :THCREATE      pid:1 tid:11
t:0x13110392 CPU:03 THREAD  :THRECEIVE     pid:1 tid:11
t:0x1311084d CPU:01 PROCESS :PROCCREATE_NAME
                      ppid:1
                       pid:2
                      name:proc/boot/slogger
t:0x13110c13 CPU:03 THREAD  :THCREATE      pid:2 tid:1
t:0x13110ce0 CPU:03 THREAD  :THRECEIVE     pid:2 tid:1
```

Exit status:

-1

An error occurred.

Errors:

Severe errors cause `traceprinter` to terminate; noncatastrophic errors are displayed in verbose mode (if the -v option is set).

Caveats:

You must run `tracelogger` before running `traceprinter`. The `tracelogger` utility creates an event file containing trace data; `traceprinter` parses and prints the data in this file.

Copyright © 2014, QNX Software Systems Limited

traceroute

Print the route packets take to network host

Syntax:

```
traceroute [-DdFIlnPrvx] [-a | -A as_server] [-f first_ttl]
           [-g gateway] [-i iface] [-m max_ttl]
           [-p port] [-q nqueries] [-s src_addr]
           [-t tos] [-w wait_time] host [packetsize]
```

Runs on:

QNX Neutrino

Options:

-A *as_server*

Turn on AS lookups and use the given server instead of the default.

-a

Turn on AS lookups for each hop encountered.

-D

Dump the packet data to standard error before transmitting it.

-d

Turn on socket-level debugging.

-F

Set the “don't fragment” bit.

-f *first_ttl*

Set the initial time-to-live used in the first outgoing probe packet.

-g *gateway*

Specify a loose source route gateway (8 maximum).

-I

Use ICMP ECHO instead of UDP datagrams.

-i *iface*

Copyright © 2014, QNX Software Systems Limited

Specify a network interface to obtain the source IP address for outgoing probe packets. This is normally only useful on a multi-homed host. (See -s for an alternative way to do this.)

-l

("el") Display the TTL (time-to-live) value of the returned packet. This is useful for checking for asymmetric routing.

-m ***max_ttl***

Set the maximum TTL (maximum number of hops) used in outgoing probe packets. The default is 30 hops (the same default as is used for TCP connections).

-n

Print hop addresses numerically only. By default, addresses are printed both symbolically and numerically. This option saves a nameserver address-to-name lookup for each gateway found on the path.

-P

Set the "don't fragment" bit and use the next hop MTU each time a "need fragmentation" error is received, thus probing the path MTU.

-p ***port***

The base UDP port number to be used in probes (default is 33434). The `traceroute` utility hopes that nothing is listening on UDP ports *base* to *base* + *nhops* - 1 at the destination host (so an ICMP PORT_UNREACHABLE message is returned to terminate the route tracing). If something is listening on a port in the default range, you can use this option to pick an unused port range.

-q ***nqueries***

The number of probes per *ttl* to *nqueries* (default is three probes).

-r

Bypass the normal routing tables and send directly to a host on an attached network. If the host isn't on a directly attached network, an error is returned. You can use this option to "ping" a local host through an interface that has no route through it (for example, after the interface was dropped by `routed`).

-s ***src_addr***

The IP address (must be given as an IP number, not a hostname) to be used as the source address in outgoing probe packets. If the host has more than

Copyright © 2014, QNX Software Systems Limited

one IP address, you can use this option to force the source address to be something other than the IP address of the interface that the probe packet is sent on. If the IP address you specify isn't one of this machine's interface addresses, an error is returned and nothing is sent.

-t *tos*

The type-of-service (TOS) to be used in probe packets (default is zero). The value must be a decimal integer in the range 0 to 255. You can use this option to see if different TOSs result in different paths.

Not all TOS values are legal or meaningful. You should find the values -t 16 (low delay) and -t 8 (high throughput) useful.

-v

Be verbose. Received ICMP packets other than TIME_EXCEEDED and UNREACHABLEs are listed.

-w *wait_time*

The time (in seconds) to wait for a response to a probe (default is 5).

-x

Toggle checksums. Normally, this prevents `traceroute` from calculating checksums. In some cases, the operating system can overwrite parts of the outgoing packet but not recalculate the checksum (so in some cases the default is to not calculate checksums and using -x causes them to be calcualted). Note that checksums are usually required for the last hop when using ICMP ECHO probes (-I).

host

The destination hostname or IP number.

packetsize

The probe datagram length (default is 40 bytes).

Description:

The Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracing the route your packets follow — or finding the gateway that's discarding your packets — can be difficult. The `traceroute` utility uses the IP protocol “time-to-live” field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to a host.

This utility attempts to trace the route an IP packet follows to an Internet host, by launching UDP probe packets with a small *ttl* (time to live) and then listening for an

Copyright © 2014, QNX Software Systems Limited

ICMP TIME_EXCEEDED reply from a gateway. Probes are started with a TTL of one and increase by one until an ICMP PORT_UNREACHABLE — which means you got to the host — is encountered or a maximum is reached. By default, this maximum is 30 hops; you can change it with the -m option.

Three probes (you can change the number with the -q option) are sent at each TTL setting and a line is printed showing the TTL, the address of the gateway, and the roundtrip time of each probe. If the answers come from different gateways, the address of each responding system is printed. If there's no response within a 5-second timeout interval (which you can change with the -w option), a * is printed for that probe.

Since the destination host shouldn't process the UDP probe packets, the destination port is set to an unlikely value. If someone on the destination is using that value, you can change it with -p.

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *mkefs* (p. 1209), *mketfs* (p. 1219), or *mkifs* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Here's a sample use and output:

```
% traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet
1  helios.ee.lbl.gov (128.3.112.1)  19 ms  19 ms  0 ms
2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  39 ms
5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  39 ms  39 ms  39 ms
6  128.32.197.4 (128.32.197.4)  40 ms  59 ms  59 ms
7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  59 ms
8  129.140.70.13 (129.140.70.13)  99 ms  99 ms  80 ms
9  129.140.71.6 (129.140.71.6)  139 ms  239 ms  319 ms
10  129.140.81.7 (129.140.81.7)  220 ms  199 ms  199 ms
11  nic.merit.edu (35.1.1.48)  239 ms  239 ms  239 ms
```

Note that lines 2 and 3 are the same. This is due to a buggy kernel on the second hop system (`lbl-csam.arpa`), that forwards packets with a zero TTL (a bug in the distributed version of 4.3 BSD). Note that you have to guess what path the packets are taking cross-country since the NSFNet (129.140) doesn't supply address-to-name translations for its NSSs.

This example is more interesting:

```
% traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
12  * * *
```


Copyright © 2014, QNX Software Systems Limited

```
13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms
```

The gateways that are 12, 14, 15, 16, and 17 hops away either don't send ICMP "time exceeded" messages or send the messages with a TTL that's too small to reach you.

Gateways 14 to 17 are running the MIT C Gateway code that doesn't send "time exceeded"s. The silent gateway 12 may be the result of a bug in the 4.[23] BSD network code (and its derivatives): versions 4.3 or earlier send an unreachable message using whatever TTL remains in the original datagram. Since for gateways the remaining TTL is zero, the ICMP time exceeded is guaranteed to *not* make it back to you. The behavior of this bug is slightly more interesting when it appears on the destination system:

```
1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms
3  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms
4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms
5  ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms
6  csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms
7  * * *
8  * * *
9  * * *
10  * * *
11  * * *
12  * * *
13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms !
```

Notice that there are 12 "gateways" (13 is the final destination) and that exactly the last half of them are missing. What's really happening is that `rip` (a Sun-3 running Sun OS3.5) is using the TTL from your arriving datagram as the TTL in its ICMP reply. So, the reply timeouts on the return path (with no notice sent to anyone since ICMPs aren't sent for ICMPs) until you probe with a TTL that's at least twice the path length. That is, `rip` is really only 7 hops away. A reply that returns with a TTL of 1 is a clue that this problem exists.

The `traceroute` utility prints a `!` after the time if the TTL is less than or equal to 1. Since vendors ship a lot of obsolete (DECs Ultrix, Sun3.*x*) or nonstandard (HPUX) software, expect to see this problem frequently and/or take care when picking the target host of your probes. Other possible annotations after the time are:

!F

Fragmentation needed.

!H

Host unreachable.

!N

Network unreachable.

Copyright © 2014, QNX Software Systems Limited

!<N>

ICMP unreachable code N.

!P

Protocol unreachable.

!S

Source route failed.

!X

Communication prohibited by the administrator.

Neither `!S` nor `!F` should ever occur — the associated gateway is broken if you see one. If almost all the probes result in some kind of unreachable, `traceroute` gives up and exits.

Intended for use in network testing, measurement, and management, `traceroute` should be used primarily for manual fault isolation. Because of the load it could impose on the network, you shouldn't use `traceroute` during normal operations or from automated scripts.

Files:

The `traceroute` utility requires the `libsocket.so` shared library.

Contributing author:

Implemented by Van Jacobson from a suggestion by Steve Deering. Debugged by others with suggestions and fixes from C. Philip Wood, Tim Seaver, and Ken Adelman.

Copyright © 2014, QNX Software Systems Limited

traceroute6

Print the route IPv6 packets take to the destination

Syntax:

```
traceroute6 [-dIlnrv] [-f firsthop] [-g gateway]
            [-m hoplimit] [-p port] [-q probes] [-s src]
            [-w waittime] target [datalen]
```

Runs on:

QNX Neutrino

Options:

-d

Enable debugging.

-f *firsthop*

Specify how many hops to skip in the trace.

-g *gateway*

Specify the intermediate gateway (`traceroute6` uses the routing header).

-I

Use ICMP ECHO instead of UDP datagrams.

-l

("el") Print both the host hostnames and numeric addresses (normally, only hostnames are printed; or, if -n is specified, only numeric addresses).

-m *hoplimit*

Specify the maximum hoplimit.

-n

Don't resolve the numeric address to a hostname.

-p *port*

Set the UDP port number to *port*.

-q *probes*

Copyright © 2014, QNX Software Systems Limited

Set the number of probes per hop count to *probes*.

-r

Bypass the normal routing tables and send directly to a host on an attached network. If the host isn't on a directly attached network, an error is returned. You can use this option to "ping" a local host through an interface that has no route through it (for example, after the interface was dropped by `routed`).

-s *src*

Use this source IPv6 address.

-v

Be verbose.

-w *waittime*

Specify the delay time between *probes*.

target

The destination hostname or IP number.

datalen

Increase the packet size by this amount. By default, the size is zero and no data is sent.

Description:

This utility prints the route that the IPv6 packets take to the destination. For more information, see *`traceroute`* (p. 1985).

This utility needs to have the setuid ("set user ID") bit set in its permissions. If you use *`mkefs`* (p. 1209), *`mketfs`* (p. 1219), or *`mkifs`* (p. 1241) on a Windows host to include this utility in an image, use the `perms` attribute to specify its permissions explicitly, and the `uid` and `gid` attributes to set the ownership correctly.

Exit status:

0

Successful completion.

Nonzero

An error occurred.

Copyright © 2014, QNX Software Systems Limited

true

Return true value (POSIX)

Syntax:

```
true
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

None.

Description:

The `true` utility does nothing but exit immediately with a zero exit code.

The `true` utility is typically used in shell scripts. One common application is to create an infinite loop, as in:

```
while true
do
   myprogram
   sleep 300
done
```

which runs `myprogram` every five minutes (300 seconds) until the shell script is interrupted.

The shell has a builtin *true* (p. 1076) command; see *ksh* (p. 1029). To make sure you use the executable, specify the full path.

Exit status:

0

Copyright © 2014, QNX Software Systems Limited

tsort

Perform a topological sort of a directed graph (POSIX)

Syntax:

```
tsort [-l] [-q] [file]
```

Runs on:

QNX Neutrino

Options:

-l

Search for and display the longest cycle. This can take a very long time.

-q

Don't display informational messages about cycles. This is primarily intended for building libraries, where optimal ordering isn't critical, and cycles occur often.

Description:

The `tsort` utility takes a list of pairs of node names representing directed arcs in a graph and prints the nodes in topological order on standard output. Input is taken from the named file, or from standard input if no file is given.

Node names in the input are separated by white space, and there must be an even number of node names.

The presence of a node in a graph can be represented by an arc from the node to itself. This is useful when a node isn't connected to any other nodes.

If the graph contains a cycle (and therefore cannot be properly sorted), one of the arcs in the cycle is ignored, and the sort continues. Cycles are reported on standard error.

Contributing author:

NetBSD

Copyright © 2014, QNX Software Systems Limited

tty

Return the user's terminal name (POSIX)

Syntax:

```
tty [-s]
```

Runs on:

QNX Neutrino

Options:

-s

Be silent; don't output the terminal name. This option is useful if you're concerned only with `tty`'s exit status.

Description:

The `tty` utility writes to the standard output the name of the terminal that's open as standard input. If standard input isn't a terminal (e.g. it's a file such as `/dev/null`), the string `not a tty` is output instead.

The -s option is deprecated by POSIX; you can achieve the effect of this option, simply and portably, by redirecting output to `/dev/null` or by using the shell builtin, *`test -c`* (p. 1071).

Examples:

The following command prints `/dev/con1` if run on console 1:

```
tty
```

The following command prints `not a tty` because `/dev/null` causes *isatty()* to return 0:

```
tty </dev/null
```

Exit status:

0

Success.

> 0

Copyright © 2014, QNX Software Systems Limited

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 22
U

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

The following are described elsewhere:

For information about:	See:
`uncompress`	*`gunzip`* (p. 920)
`usblauncher`	*Device Publishers Developer's Guide*
`usbpub`	*Device Publishers Developer's Guide*

This chapter describes the utilities, etc. whose names start with “U”.

Copyright © 2014, QNX Software Systems Limited

uesh

Micro-embedded shell (QNX Neutrino)

Syntax:

```
uesh [script_file]
```

Runs on:

QNX Neutrino

Options:

script_file

A file containing shell commands to execute.

Description:

The uesh utility provides a subset of the functionality found in the standard embedded shell, *esh* (p. 705). You should find uesh useful for situations where memory requirements are limited. For example, you could use it to run a simple system initialization file for an embedded system.

The micro-embedded shell has some very significant limitations:

- no pipes
- no aliases
- no filename- or command-completion
- no `set` command

For applications that require greater functionality, use *esh* (p. 705) or the full shell, *sh* (p. 1760).

Command-line format

In uesh, command lines take this form:

command arg1 arg2 ... [*redir-op file*] [&]

Where:

command

The command to be executed. If it doesn't start with a slash, the command follows the path set by the ***PATH*** environment variable.

Copyright © 2014, QNX Software Systems Limited

redir-op file

A redirection operator. When a command is invoked, three standard files are set up in its environment. These files, *standard input*, *standard output*, and *standard error* output (*stdin*, *stdout*, *stderr*), are usually attached to the active terminal. You can redirect a command's standard input, standard output, and standard error as follows:

Specifying:	Will:
<*file*	redirect standard input from this file.
>*file*	Redirect standard output to this file. If the file exists, it's overwritten; if the file doesn't exist, it's created.
>>*file*	Redirect standard output to this file. If the file exists, the information is appended to the end of the file; if the file doesn't exist, it's created.
2>*file*	Do the same as >*file*, but for standard error.
2>>*file*	Do the same as >>*file*, but for standard error.
&	If a command contains an unquoted &, then `uesh` doesn't wait for the command to complete execution but immediately moves on to process the next command. The standard input of the command is redirected from `/dev/null`, and `SIGINT` and `SIGQUIT` are ignored.

Filename expansion

The `uesh` shell doesn't support filename expansion. Such shorthands as `*.c` for all files ended in `.c` don't work.

Quoting

The following characters have a special meaning in `uesh`:

```
&  \  "  [  space
```

To suppress the special meaning of these characters and keep their literal meaning, use *quoting*.

Copyright © 2014, QNX Software Systems Limited

To quote a sequence of characters or sequence of words, enclose the sequence in double quotes. To quote a single character, use double quotes or precede it with the escape character (\).

Escape character (backslash)

The escape character (\) preserves the literal meaning of the next character. You can't obtain a single backslash by quoting \ with double quotes. To obtain a backslash, enter \\ instead.

Double quotes

Enclosing characters and words in double quotes ("") preserves the literal meaning of all characters within double quotes, with the exception of the \ character. For example:

```
"ab cd"
```

represents a single, five-character argument.

You can keep the literal meaning of a double quote with the \ character. For example:

```
ab\"cd
```

represents the single, five-character argument ab"cd.

Builtin commands

The following commands are built into uesh (that is, uesh interprets and executes them internally):

- *cd* (p. 2000)
- *emount* (p. 2000)
- *ewaitfor* (p. 2001)
- *exec* (p. 2002)
- *exit* (p. 2002)
- *export* (p. 2002)

cd command

```
cd [directory]
```

Change the working directory of the current execution environment. If *directory* isn't specified, the value of the ***HOME*** environment variable becomes the new working directory.

emount command

```
emount special directory [fs_type]
```

Mount a special device. The arguments are:

Copyright © 2014, QNX Software Systems Limited

special

The name of the special device.

mountpoint

Where to mount the device on your system.

type

The type of filesystem or manager to mount:

type:	Filesystem or manager:
cd	*fs-cd.so* (p. 787)
cifs	*fs-cifs* (p. 790)
dos	*fs-dos.so* (p. 795)
ext2	*fs-ext2.so* (p. 807)
mac	*fs-mac.so* (p. 809)
nfs	*fs-nfs2* (p. 811), *fs-nfs3* (p. 814)
nt	*fs-nt.so* (p. 818)
qnx4	*fs-qnx4.so* (p. 820)
qnx6	*fs-qnx6.so* (p. 823)
udf	*fs-udf.so* (p. 829)

The default is qnx4.

The `emount` command was implemented in QNX Momentics 6.3.0 Service Pack 2.

`ewaitfor` command

```
ewaitfor path [max_seconds [delay]]
```

Wait until the given path exists. The arguments are:

path

The path to test.

max_seconds

The maximum number of seconds to wait for the file to appear. The default is 1 second.

Copyright © 2014, QNX Software Systems Limited

delay

The number of milliseconds to wait between attempts. The default is 100 ms.

The `ewaitfor` command was implemented in QNX Momentics 6.3.0 Service Pack 2.

exec command

```
exec [command [argument...]]
```

Execute a command and/or manipulate file descriptors.

The `exec` command opens, closes, or copies file descriptors as specified by any I/O redirections given as part of *argument*. If a command is specified, that command is spawned as a replacement for `uesh`. Any specified arguments are passed to the spawned process.

exit command

```
exit [n]
```

Cause `uesh` to exit with the exit status specified by *n*. If *n* isn't specified, `uesh` exits with the status of the last command executed.

export command

```
export name[=word]...
export -p
```

Mark environment variables for export, which causes them to be in the environment of subsequently executed commands. If you specify the -p option, the names and values of all exported variables are written to the standard output.

Examples:

Invoke `uesh`:

```
uesh
```

Environment variables:

HOME

The pathname of the user's home directory

LOGNAME

The user's *login* (p. 1121) name.

PATH

Copyright © 2014, QNX Software Systems Limited

The directory search path used by uesh for locating executable programs. To change ***PATH***, you must use the `export` command.

If ***PATH*** isn't in the existing environment when `uesh` is invoked, it's set to `/bin:/usr/bin`. For more information on setting ***PATH***, see "Setting ***PATH*** and ***LD_LIBRARY_PATH***" in the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*.

SHELL

The pathname of the user's preferred shell.

TERM

The terminal type.

TMPDIR

The pathname of a directory where utilities can create temporary files.

TZ

The timezone setting.

Copyright © 2014, QNX Software Systems Limited

ulink_ctrl

Control a USB DCD link

Syntax:

```
ulink_ctrl [-l 0|1] [-s name] [-w num]
```

Runs on:

QNX Neutrino

Options:

-l 0|1

("el") Set the USB link state to disconnected (0) or connected (1).

-s *stack*

The name of the USB manager to query (e.g., `/dev/io-usb-dcd/io-usb`).

-w *num*

Wait *num* seconds for the USBDC stack (default 60 seconds).

Description:

The `ulink_ctrl` utility controls a USB cable connection as seen by the USB host.

Examples:

Allow the USB host to see the USBDC connection:

```
ulink_ctrl -l1
```


Copyright © 2014, QNX Software Systems Limited

umask

Get or set the file mode creation mask (POSIX)

Syntax:

```
umask [-o|-s|mask]
```

Runs on:

QNX Neutrino

Options:

-o

Display the current mask, in octal.

-s

Display the current mask in symbolic form. This is the default output.

mask

Set the file mode creation mask to *mask*, which you can specify either as an octal number or as a symbolic representation.

If you specify the *mask* in octal form, it replaces the current file mode creation mask. Every bit that's set describes a mode bit that *won't* be allowed in the file mode of created files. In other words, it says: “mask this bit `off`.”

The symbolic form of the *mask* is an expression that modifies or replaces the current file mode creation mask. The form of the symbolic *mask* is similar to that of the mode operand for the *chmod* (p. 124) utility:

```
[[augo] [+|-|=] [rwx]] [,symbolic_mask]
```

Where:

a

User, group, and other access.

u

User access.

g

Copyright © 2014, QNX Software Systems Limited

Group access.

o

Other access.

+

Add these permissions to the current mask.

-

Remove these permissions from the current mask.

=

Replace the current mask with these permissions.

r

Read permission.

w

Write permission.

x

Execute permission.

Once the symbolic mask expression has been applied to the current file mode creation mask, any occurrence of [r,w, x] describes a mode bit that *is* allowed in the file mode of created files. The absence of a symbol means that permission isn't allowed and is masked "off."

Description:

The `umask` utility sets the file mode creation mask of the invoking process to the value specified by the *mask* operand. The file mode creation mask affects the initial value of the file permission bits of subsequently created files when no mode is specified.

When files are created without specifying the permission mode bits, the filesystem assigns default permissions of `0777` (`rwxrwxrwx`) to directories and executable files, thereby giving read, write, and execute privileges for user, group, and others. For files that aren't executable, permissions of `0666` (`rw-rw-rw-`) are assigned. The `umask` utility is used to adjust these defaults.

The file mode creation mask is inherited by any children of the current process.

You can use either of the display forms (-o or -s) as the *mask* operand to a subsequent invocation of `umask`.

Copyright © 2014, QNX Software Systems Limited

As in the *chmod* (p. 124) utility, the use of the octal number form of mask values is deprecated.

The shell has a builtin *umask* (p. 1080) command; see *ksh* (p. 1029). To make sure you use the executable, specify the full path.

Examples:

1. Set mask to allow read, write, and execute, for user, group, and others:

```
$ umask a=rwx
```

   Display the current file creation mode mask in symbolic form:

```
$ umask -s
u=rwx, g=rwx, o=rwx
```

   Display the current file creation mode mask in octal:

```
$ umask -o
00
```

2. Specify that no permissions for group and others be allowed; set the mask to allow only read and write for user only:

```
$ umask u=rw
```

   Display the current file creation mode mask in symbolic form:

```
$ umask
u=rw,g=,o=
```

   Display the current file creation mode mask in octal:

```
$ umask -o
0177
```

3. Add read permissions for group and others:

```
$ umask go+r
```

   Display the current file creation mode mask in symbolic form:

```
$ umask
u=rw,g=r,o=r
```

   Display the current file creation mode mask in octal:

```
$ umask -o
```

Copyright © 2014, QNX Software Systems Limited

```
0133
```

Exit status:

0

The file mode creation mask was successfully changed, or no *mask* operand was supplied.

>0

An error occurred. The process's file mode creation mask isn't changed.

Copyright © 2014, QNX Software Systems Limited

umount

Unmount a device

Syntax:

```
umount [-f] path [path ...]
```

Runs on:

QNX Neutrino

Options:

-f

Force an unmount, even if the device is busy.

path

The root mountpoint to be unmounted.

Description:

This utility unmounts the devices that were mounted as the given paths.

If a device is busy (for example, if a file is open), `umount` fails. You can force the device to be unmounted by specifying the -f option.

Copyright © 2014, QNX Software Systems Limited

uname

Return the name of the operating system (POSIX)

Syntax:

```
uname [-amnprsv] [-S name]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Behave as if the options -snrvmp were specified.

-m

Write the name of the hardware type on which the system is running.

-n

Write the name of this node.

-p

Write the processor name.

-r

Write the current release level of the operating system (indicated by a number).

-S *name*

Set the host name.

-s

Write the name of the operating system.

-v

Write the current version level of this release of the operating system (indicated by a timestamp).

Copyright © 2014, QNX Software Systems Limited

Description:

The uname utility writes to standard output information on the name and release of the operating system being run. A portable application may use uname on any POSIX system to determine what operating system it's running under.

If you don't specify any options, uname writes the operating system name (QNX).

Examples:

Write the operating system name:

```
uname
```

Write a formatted string showing the name, release level, and version level of the operating system:

```
printf "OS: %s release %s version %s\n" $(uname -srv)
```

Exit status:

0

Success

> 0

An error occurred.

Caveats:

The *pidin* (p. 1521) utility provides more detailed information than uname, but pidin is a QNX Neutrino utility and isn't present on other systems.

Copyright © 2014, QNX Software Systems Limited

unexpand

Convert spaces to tabs (POSIX)

Syntax:

```
unexpand [-a] [-t tabsize] [file...]
```

Runs on:

QNX Neutrino

Options:

-a

In addition to the default behavior of replacing leading spaces, translate two to eight consecutive spaces immediately preceding a tab stop into a tab. A tab stop is a column position that's a multiple of eight column positions.

-t *tabsize*

(QNX Neutrino extension) Set the tab stops *tabsize* columns apart (default is 8). The *tabsize* argument consists of a single positive decimal integer.

file

The pathname of a text file to be used as input.

Description:

The unexpand utility copies files or the standard input to the standard output, translating each group of eight spaces at the beginning of a line into a tab character. Any backspace characters in the input are copied to the output, and each causes the column position count for tab calculations to be decremented; the count is never decremented below zero.

Examples:

For the file sourcecode, convert every run of eight spaces at the beginning of a line into a single tab:

```
unexpand sourcecode
```

Convert every run of two to eight spaces that precedes a tab stop into a single tab:

```
unexpand -a sourcecode
```


Copyright © 2014, QNX Software Systems Limited

Convert every run of two to four spaces that precedes a tab stop into a single tab:

```
unexpand -a -t4 sourcecode
```

Exit status:

0

Successful completion.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

unifdef

Remove `ifdef`'ed C/C++ lines

Syntax:

```
unifdef [-l] [-t] [-c] [[-Dsym] [-Usym] [-iDsym] [-iUsym]]...
 [file]
```

Runs on:

QNX Neutrino

Options:

-D*sym* -U*sym*

Specify which symbols to define or undefine respectively. The lines inside those `ifdef`s are copied to the output or removed as appropriate. The `ifdef`, `ifndef`, `else`, and `endif` lines associated with *sym* are also removed. If an `ifdef X` occurs nested inside another `ifdef X`, then the inside `ifdef` is treated as if it were an unrecognized symbol.

-c

Cause the operation to be complemented, i.e. the lines that would have been removed or blanked are retained and vice versa.

-l

Replace removed lines with blank lines instead of deleting them.

-t

Disable parsing for C/C++ comments and quotes, which is useful for plain text.

-iD*sym* -iU*sym*

Ignore the specified `ifdef`s. Specifies which `ifdef` symbols to parse or not to parse for quotes and comments inside respectively. Parsing is done by default, and the -t option overrides these options.

Description:

The `unifdef` utility removes `ifdef`'ed lines from C or C++ code. You must specify at least one of -D, -U, -iD, and -iU. This utility copies output to *stdout*, and takes its input from *stdin* if you don't specify a *file* argument.

Copyright © 2014, QNX Software Systems Limited

uniq

Report or filter out repeated lines in a file (POSIX)

Syntax:

```
uniq [-c] [-d|-u] [-f fields] [-s chars]
     [input_file [output_file]]
```

Deprecated syntax:

```
uniq [-c] [-d|-u] [-n] [+m]
     [input_file [output_file]]
```

Runs on:

QNX Neutrino

Options:

-n

(deprecated, replaced by -f) Ignore the first *n* fields when doing comparisons, where *n* is a number.

+m

(deprecated, replaced by -s) Ignore the first *m* characters when doing comparisons, where *m* is a number.

-c

Precede each output line with the number of times the line occurred in the input.

-d

Suppress the writing of lines that aren't repeated in the input.

-f *fields*

Ignore the first *fields* on each input line when doing comparisons, where *fields* is a positive decimal integer. A field is a string of nonblank characters separated from adjacent fields by blanks.

-s *chars*

Ignore the first *chars* characters when doing comparisons, where *chars* is a positive decimal integer. If specified in conjunction with the -f option, the first *chars* characters after the first *fields* fields are ignored.

Copyright © 2014, QNX Software Systems Limited

-u

Suppress the writing of lines that are repeated in the input.

input_file

The pathname of the input file. If you don't specify any input files, the standard input is used.

output_file

The pathname of the output file. This name must differ from the name of the input file. If you don't specify an output file, the standard output is used.

Description:

The `uniq` utility reads an input text file, comparing adjacent lines, and writes one copy of each input line to the output. The second and succeeding copies of repeated adjacent input lines aren't written.

To obtain a report of unique lines in a file, the input file *must* be sorted prior to running `uniq`.

Examples:

Look for repeated adjacent lines in `datfile`:

```
uniq datfile
```

Environment variables:

LC_TYPE

The locale for character classification, used to determine the characters constituting a blank in the current locale.

QNX Neutrino currently supports only the POSIX (i.e. C) locale.

Exit status:

0

Success

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Errors:

If *output_file* is created, it isn't removed when an error occurs.

Copyright © 2014, QNX Software Systems Limited

unlink

Call the unlink() function to delete a file

Syntax:

`unlink` *file*

Runs on:

QNX Neutrino

Options:

file

The pathname of an existing file.

Description:

The `unlink` utility is a command-line interface to the *unlink()* function:

```
(void)unlink( file );
```

You need the appropriate permissions (typically write permission in the directory that contains *file*) in order to use the `unlink` utility. In order to unlink a directory, you need the appropriate permissions, and the filesystem must also allow it (see `_PC_LINK_DIR` in the description of *pathconf()* in the QNX Neutrino *C Library Reference*).

Exit status:

0

Successful completion.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

unzip

Extract files from a zip archive

Syntax:

```
unzip [-Z] [-opts[modifiers]] file[.zip] [list]
      [-x xlist] [-d exdir]
```

Runs on:

QNX Neutrino, Microsoft Windows

Targets:

x86 only

Options:

The options include:

-d *exdir*

Extract files into *exdir*.

-f

Freshen the existing files.

-l

("el") List archive files (short format).

-P *password*

Use the given password to unencrypt the archive entries.

Specifying a plain-text password on the command line or in a script can be a security problem.

-p

Extract files to pipe (*stdout*).

-T

Update the timestamp for the archive to match the latest timestamp of the archive files.

Copyright © 2014, QNX Software Systems Limited

-t

Test the archive files.

-u

Update existing files and create new ones if needed.

-v

Be verbose or print diagnostic version information.

-x *xlist*

Exclude the files in the *xlist*.

-Z

ZipInfo mode. If the first option on the command line is -Z, the remaining options are taken as ZipInfo options.

-z

Display the archive comment only.

The modifiers include:

-a

Automatically convert any text files.

-aa

Extract all files as text files.

-C

Match files case-insensitively.

-j

Junk the path to the file; do not make directories.

-K

Keep setuid/setgid/tacky permissions.

-L

Convert some names to lowercase.

-M

Pipe all output through an internal pager that's similar to the `more` command.

-n

Copyright © 2014, QNX Software Systems Limited

Never overwrite existing files.

-o

Overwrite existing files *without* prompting.

-q

Perform operations quietly. Use -qq to make them quieter.

-U

Use escapes for all non-ASCII Unicode.

-UU

Ignore any Unicode fields.

-V

Retain VMS file version numbers.

-X

Restore user and group (UID/GID) information.

Description:

The `unzip` utility lists, tests, or extracts files from a `ZIP` archive. The default behavior (with no options) is to extract into the current directory (and subdirectories below it) all files from the specified `ZIP` archive. A companion program, *`zip`* (p. 2081), creates `ZIP` archives; both programs are compatible with archives created by PKZIP and PKUNZIP.

Examples:

Extract all members of the archive `letters.zip` into the current directory and subdirectories below it, creating any subdirectories as necessary, using `unzip`:

```
unzip letters
```

Extract all members of `letters.zip` into the current directory only:

```
unzip -j letters
```

Test `letters.zip`. Use the following command to print only a summary message indicating whether the archive is all right or not:

```
unzip -tq letters
```

Test all zipfiles in the current directory, and print only the summaries:

```
unzip -tq \*.zip
```

Copyright © 2014, QNX Software Systems Limited

Extract all members of `letters.zip` whose names end in *.tex*, to standard output, auto-convert to the local end-of-line convention, and pipe the output into `more`:

```
unzip -ca letters \*.tex | more
```

Extract the binary file `paper1.dvi` to standard output, and pipe it to a printing program:

```
unzip -p articles paper1.dvi | dvips
```

Extract newer versions of the files already in the current directory and create any files not already there:

```
unzip -uo sources
```

Display a diagnostic screen showing which `unzip` and `zipinfo` options are stored in environment variables:

```
unzip -v
```

See whether decryption support was compiled with the files:

```
unzip -v
```

See the compiler that `unzip` used:

```
unzip -v
```

Environment variables:

UNZIP

A set of default options for `unzip`. For example:

```
export UNZIP="-qq"
```

Exit status:

0

The operation succeeded.

2

A generic error in the zipfile format was detected, but processing may have completed successfully anyway; a warning was generated in the process.

3

A severe error in the zipfile format was detected; processing probably failed immediately.

Copyright © 2014, QNX Software Systems Limited

4

unzip was unable to allocate memory for one or more buffers during program initialization.

5

unzip was unable to allocate memory or unable to obtain a tty to read the decryption password(s).

6

unzip was unable to allocate memory during decompression to disk.

9

The specified zipfiles were not found.

10

Invalid options were specified on the command line.

11

No matching files were found.

50

The disk is, or was, full during extraction.

51

The end of the ZIP archive was encountered prematurely.

80

The user aborted unzip prematurely with **Ctrl**–**C** or a similar command.

81

Testing or extraction of one or more files failed, due to unsupported compression methods or unsupported decryption.

82

No files were found, due to bad decryption password(s). If even one file is successfully processed, however, the exit status is 1.

Copyright © 2014, QNX Software Systems Limited

uptime

Display the length of time that the system has been running

Syntax:

```
uptime
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `uptime` utility displays the current local time, by the length of time that the system has been running (in days, hours, and minutes, as appropriate), and the number of users logged in.

Copyright © 2014, QNX Software Systems Limited

usb

Display USB device configuration

You must be `root` to run this utility.

Syntax:

```
usb [-b busno -d devno -N name] [-t] [-V] [-v]
```

Runs on:

QNX Neutrino

Options:

-b *busno*

Display devices on specific *busno only*.

-d *devno*

Display device *devno only*.

-N *name*

Name of the USB manager to query. The default is `/dev/io-usb/io-usb`.

-t

Use a “tree-like” output format.

-V

Display vendor-unique descriptors (at given verbosity level).

-v

Be verbose. The level of detail increases as you add `v`'s.

Description:

The `usb` utility displays USB device configuration. As you repeat the number of -v (verbose) options, you'll get more levels of detail.

Copyright © 2014, QNX Software Systems Limited

Examples:

Here's some sample output from the command `usb -vvv`. Note that two USB devices (an Ethernet card and a printer) are installed in this case.

```
USB (UHCI) v1.10, v1.00 DDK
    Control, Interrupt, Bulk, Isoch, Low, High

Device Address             : 1
Vendor                     : 0x0506 (3COM)
Product                    : 0x03e8 (3COM USB Network Interface (3C19250))
Device Release             : r0.02
USB Spec Release           : v1.00
Serial Number              : 009004BB78F9
Class                      : 0x00 (Independent per interface)
Max PacketSize0            : 8
Languages                  : 0x0409 (English)
Configurations             : 1
  Configuration            : 1
    Attributes             : 0x80 (Bus-powered)
    Max Power              : 110 mA
    Interfaces             : 1
      Interface            : 0 / 0
        Class              : 0x00 (Unknown)
        Subclass           : 0x00
        Protocol           : 0x00
        Endpoints          : 3 + Control
          Endpoint         : 0
            Attributes     : Control
            Max Packet Size: 8
          Endpoint         : 1
            Attributes     : Bulk/IN
            Max Packet Size: 64
          Endpoint         : 2
            Attributes     : Bulk/OUT
            Max Packet Size: 64
          Endpoint         : 3
            Attributes     : Interrupt/IN
            Max Packet Size: 8
            Interval       : 1 ms

Device Address             : 2
Vendor                     : 0x04a9 (Canon)
Product                    : 0x1055 (BJC-85)
Device Release             : r0.70
USB Spec Release           : v1.10
Serial Number              : 44OFBr
Class                      : 0x00 (Independent per interface)
Max PacketSize0            : 8
Languages                  : 0x0409 (English)
Configurations             : 1
  Configuration            : 1
    Attributes             : 0x40 (Self-powered)
    Max Power              : 2 mA
    Interfaces             : 1
      Interface            : 0 / 0
        Class              : 0x07 (Printer)
        Subclass           : 0x01
        Protocol           : 0x02
        Endpoints          : 2 + Control
          Endpoint         : 0
            Attributes     : Control
            Max Packet Size: 8
          Endpoint         : 1
            Attributes     : Bulk/OUT
            Max Packet Size: 64
          Endpoint         : 2
            Attributes     : Bulk/IN
            Max Packet Size: 64
```


Copyright © 2014, QNX Software Systems Limited

use

Print a usage message (QNX Neutrino)

Syntax:

```
use [-aeis] [-d directory] [-f filelist] files
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-a

Extract all usage information from the load module in its source form, suitable for piping into *usemsg* (p. 2030).

-d *directory*

Recursively display information for all files under *directory*.

-e

Include only ELF files.

-f *filelist*

Read a list of files, one per line, from the specified *filelist* file, and display information for each.

-i

Display build properties about a load module.

-s

Display the version numbers of the source used in the executable.

files

One or more executable load modules or shell scripts that contain usage messages.

Description:

The use utility displays a usage message for the specified executable programs or shell scripts.

Copyright © 2014, QNX Software Systems Limited

Regardless of your current terminal settings specified, the `use` utility automatically includes a line break after 80 characters.

The `use` utility searches for *files*, using the default command search (***PATH***), and displays the usage message (if any) that it finds in the load files or shell scripts.

If the ***LANG*** environment variable is set, a usage message of that language is displayed, if available. Alternate language usage messages are not available within shell scripts. However, it's easy to edit shell script messages. While usage messages included with standard versions of the QNX Neutrino RTOS are in English only, it's possible to add alternate language usage messages by placing the revised usage message in a separate file, and using the *usemsg* (p. 2030) utility to insert the usage message in the executable in question.

Usage messages in shell scripts

Usage messages are implemented in binary executable programs using a special form of resource record in the load modules. Usage messages are implemented in shell scripts using a format similar to that used in the C source code and interpreted by the `usemsg` utility.

In shell scripts, the `use` utility scans each line from the beginning of the script, looking for a line starting with the # character (i.e. a comment) and containing the string __USAGE. The usage message begins on the next line and consists of all subsequent lines up to, but not including, the first line that either starts with `#endif` or starts with a character other than #.

Here's a sample usage message in a shell script:

```
#ifdef __USAGE
#%C thread_id
#Where:
#  thread_id   is the thread ID you want to act on
#endif
```

If the shell script is called `foo`, and you invoke `use foo`, the following message is displayed:

```
foo thread_id
Where:
  thread_id    is the thread ID you want to act on
```

In the above shell script fragment, the message starts with:

```
#%C thread_id
```

and ends with:

```
#  thread_id   is the thread ID you want to act on
```

Within the body of the usage message, the leading #s are stripped by the `use` utility and don't form part of the message that's displayed. As with the C language usage

Copyright © 2014, QNX Software Systems Limited

message convention (see *usemsg* (p. 2030)), a %CTab at the start of a line is replaced by the program name (or filename of the shell script) and a tab character at the start of a line spaces over the same number of spaces as the last previous occurrence of %CTab.

You can place the usage message almost anywhere in most shell scripts. Placing it at the beginning results in quicker response for extracting the usage message at the expense of a very slight slowdown in execution of the shell script. If you're running a shell that doesn't recognize lines beginning with # as comments, you should place the usage message after an explicit *exit* (p. 1063).

Examples:

Display a usage message for the `ls` utility:

```
use ls
```

Copyright © 2014, QNX Software Systems Limited

usemsg

Change the usage message for a command (QNX Neutrino)

Syntax:

```
usemsg [-c] [-i id[=value]] [-f info_file] loadfile [msgfile]
```

Runs on:

QNX Neutrino, Linux, Microsoft Windows

Options:

-c

The usage message is contained in a C source program delimited by:

```
#ifdef __USAGE
...
#endif
```

Note that there are *two* underscores before USAGE.

-f *filename*

Read *filename* for lines containing *id=value* pairs to import into *loadfile*.

-i *id=value*

Add the information tag *id* to *loadfile* with the specified *value*.

value doesn't need to be specified for the DATE or NAME *id*s.

The DATE and NAME keys will be added automatically when any other key is added.

id is translated into upper-case.

-l

Use *ldrel* (p. 1098) to add the specified usage message.

-o

Use *objcopy* (p. 1406) to add the specified usage message. This is the default behavior.

Copyright © 2014, QNX Software Systems Limited

The -o option is *required* if you're running `usemsg` on a binary that has its data segment before its code segment. Without the -o option, `usemsg` will corrupt these reordered binaries.

-s *string*

Import the usage message from the `#ifdef` *string* section in a C source file. If *string* is omitted, __USAGE will be used.

If multiple -s options are specified, `usemsg` will search for them in order and use the first *string* section found.

loadfile

The name of an executable program to extract or insert a usage message. The current ***PATH*** environment variable is searched to locate *loadfile*.

msgfile

A text file or a C source file containing a usage message (see -c). If the *msgfile* name ends in `.c`, a C source is assumed. If present, this argument is used as the name of the message file to insert into the load file. If the *msgfile* argument isn't present, the usage message is read from the load file and printed to standard output.

Description:

The `usemsg` utility lets you examine or change the usage record contained within a QNX Neutrino executable program. All utilities supplied with the QNX Neutrino RTOS are shipped with a usage message describing their options. This information is kept in a resource record in the load file. Since this usage text isn't loaded into memory when the program is run, you can make it as large as 32K characters without affecting the size of your program at runtime.

The *use* (p. 2027) utility prints usage messages. For example:

```
use ls
use more
use pidin
```

Developers may use the `usemsg` utility to add usage messages to their programs.

Displaying help messages in ported executables

If you are porting or developing an executable that already has a help message invoked by an argument, you can make `use` display the existing help message by adding one extra line in the executable, like this:

```
%digit> cmd argument
```

Copyright © 2014, QNX Software Systems Limited

Where *digit* is where to read the output from, either `1` (*stdout*) or `2` (*stderr*). The `use` utility itself always prints to *stdout* but executables may print to *stdout* or *stderr*.

For example, if `some_gnu_tool` has an option --help that sends a help message to *stdout*, add a line like this:

```
%1> some_gnu_tool --help
```

or

```
%1> %C --help
```

In this example, when someone types:

```
use some_gnu_tool
```

The `use` utility spawns:

```
some_gnu_tool --help
```

and then prints the output.

If the executable sends its output to *stderr*, add this line instead:

```
%2> some_gnu_tool --help
```

Adding or changing a usage message

There are two forms of adding a usage message to a load file. One form assumes a simple text file, while the other assumes that the usage message is contained in a C source program:

```
usemsg program textfile
usemsg program program.c
```

In the second form, the C source is scanned and all text between an `#ifdef __USAGE` and the next `#endif` is used. In both cases, any existing usage message is replaced by the new message. Note that this utility lets you both change existing usage messages and add usage messages to programs that have none. You don't need the program source.

The `usemsg` utility provides a simple grammar that allows it to support usage messages in several different languages. It also supports different messages linked to the name used to invoke the usage. For example, if *less* (p. 1100) and *more* (p. 1309) are links to the same load file, they can each have their own usage within the same usage record in the file.

The grammar consists of the special symbol % in the first column followed by an action character as follows:

%%

A single %.

Copyright © 2014, QNX Software Systems Limited

%-*command*

The start of a specific command's usage message.

%=*language*

The start of a new language.

%C<tab>

Replace with name of command and a space.

<tab>

Insert spaces equal to the length of the command + 1.

To extract the entire usage message, including all languages and the grammar control sequences, name the *loadfile* and don't specify a *msgfile*.

The %-*command* and %=*language* are both optional. If both are specified, the %-*command* is followed by one or more %=*language* sections followed by another %-*command* and another set of %=*language* sections. The following examples should clarify the required nesting:

```
%C   a single language message

%=english
%C   an English language message
%=french
%C   a French language message

%-less
%C   single language message for less
%-more
%C   single language message for more

%-less
%=english
%C   an English language message for less
%=french
%C   a French language message for less
%-more
%=english
%C   an English language message for more
%=french
%C   a French language message for more
```

If multiple language usage messages are available, *use* (p. 2027) employs the ***LANG*** environment variable to select a language. If ***LANG*** isn't defined or doesn't match any language present, then the first usage message is printed. Likewise, if multiple command names are present, the command passed as an argument is used to select a command. If no match occurs, the first usage message is printed.

Copyright © 2014, QNX Software Systems Limited

Examples:

Insert a usage message from C source into `myprog`:

```
usemsg myprog myprog.c
```

Extract the entire usage message for *pidin* (p. 1521), edit the message, then reinsert the changed message:

```
usemsg pidin > my_pinitmsg
vi my_pinitmsg
usemsg pidin my_pinitmsg
```


Copyright © 2014, QNX Software Systems Limited

uud

Decode a file that was encoded with uue

Syntax:

```
uud [-n] [-d] [-s dir] [-t dir] input-file
```

Runs on:

QNX Neutrino

Options:

-n

Don't check the line sequence.

-d

Use debug (verbose) mode.

-s *dir*

The source directory for all input files. You must end the name of the directory with a directory separator (i.e. a slash).

-t *dir*

The target directory for all output files. You must end the name of the directory with a directory separator (i.e. a slash).

input-file

The name of the file to decode. If *input-file* is `-`, read from standard input.

Description:

The uud utility decodes a file that was encoded using *uue* (p. 2037).

If the filename is in the form *input-file*`.uaa` (i.e. you used the -n to uue to break a file into chunks), uud automatically includes the subsequent pieces.

Copyright © 2014, QNX Software Systems Limited

uudecode

Decode a file that was encoded with uuencode *(POSIX)*

Syntax:

```
uudecode [-p | -o outfile] [file ...]
```

Runs on:

QNX Neutrino

Options:

-o *outfile*

Decode to *outfile*; otherwise uudecode recreates the original file.

-p

Decode the file to *stdout*; otherwise uudecode recreates the original file.

file

The file(s) to decode; the default is to decode from *stdin*.

Description:

The uudecode utility decodes a file that was encoded using *uuencode* (p. 2038).

Copyright © 2014, QNX Software Systems Limited

uue

Encode a binary file into ASCII

Syntax:

```
uue [-n] input_file [-]
```

Runs on:

QNX Neutrino

Options:

-n

The number of lines to put into each file.

input_file

The file that you want to encode.

Description:

The uue utility encodes a binary file as ASCII (e.g. for mailing). The encoding takes 3 bytes (24 bits) from the input file and renders them as 4 bytes in the output file.

By default, the output file is *input_file*.uue; if you specify a dash at the end of the command, uue writes the encoded file to standard output.

If you want to read from standard input (e.g. a pipe), use *uuencode* (p. 2038) instead of uue.

If you specify the -n option, uue writes the encoded file to *input_file*.uaa, *input_file*.uab, and so on.

To decode the file, use *uud* (p. 2035).

Copyright © 2014, QNX Software Systems Limited

uuencode

Encode a binary file or standard input into ASCII (POSIX)

Syntax:

```
uuencode [-m] [file] name
```

Runs on:

QNX Neutrino

Options:

-m

Use the MIME Base64 encoding; otherwise use the uuencode historical algorithm.

file

The file that you want to encode. If you don't specify a file, uuencode reads from standard input.

name

The name to embed in the encoded file for use by *uudecode* (p. 2036).

Description:

The uuencode utility encodes a binary file as ASCII (e.g. for mailing). The encoding takes 3 bytes (24 bits) from the input file and renders them as 4 bytes in the output file.

By default, uuencode writes the encoded file to standard output. If you don't specify an input file, uuencode reads from standard input (unlike *uue* (p. 2037)).

To decode the file, use *uudecode* (p. 2036).

Copyright © 2014, QNX Software Systems Limited

Chapter 23
V

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “V”.

Copyright © 2014, QNX Software Systems Limited

vi

Visual interface editor clone (UNIX)

Syntax:

```
vi [options]... [+command] file...
```

Runs on:

QNX Neutrino

Options:

See *elvis* (p. 672) for a full listing.

Description:

The `vi` utility is an interactive fullscreen editor that's compatible with the Unix/POSIX `vi` editor. The `vi` utility in the QNX Neutrino RTOS is a link to `elvis`.

For more information, see Linda Lamb, *Learning the vi Editor*, O'Reilly and Associates, 1990.

Contributing author:

Steve Kirkendall

Copyright © 2014, QNX Software Systems Limited

view

Visual interface editor clone (UNIX)

Syntax:

```
view [options]... [+command] file...
```

Runs on:

QNX Neutrino

Options:

See *elvis* (p. 672) for a full listing.

Description:

The `view` utility is an interactive fullscreen editor that's compatible with the Unix/POSIX `vi` editor. The `view` utility in the QNX Neutrino RTOS is a link to `elvis`.

For more information, see Linda Lamb, *Learning the vi Editor*, O'Reilly and Associates, 1990.

Contributing author:

Steve Kirkendall

Copyright © 2014, QNX Software Systems Limited

Chapter 24
W

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “W”.

Copyright © 2014, QNX Software Systems Limited

waitfor

Wait until a path exists

Syntax:

```
waitfor pathname [wait_time[:poll_ms]]
```

Runs on:

QNX Neutrino

Options:

pathname

The path to test.

wait_time

The maximum number of seconds to wait for the file to appear. It can include one decimal digit to specify tenths of a second. The default is 5.0 seconds.

poll_ms

(QNX Neutrino 6.6 or later) The number of milliseconds to wait between checks for the path. The default is 100 milliseconds.

Description:

The `waitfor` utility pauses temporarily until a *stat()* on the specified *pathname* succeeds. It's often used for synchronization, to allow a resource manager to perform its startup functionality, and *then* for the process manager to proceed with the further interpretation of the script file.

Native *io-pkt* (p. 1007) and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a `waitfor` command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/en0`, use `if_up -p en0`.

Copyright © 2014, QNX Software Systems Limited

wave

Play back audio data

Syntax:

```
wave [options]
```

Runs on:

QNX Neutrino

Options:

-a [*card_num*:]*dev_num*

The card and device number to play out on.

-c *hw_channel_bitmask*[,*hw_channel_bitmask*...]

Voice matrix configuration. The first argument is the hardware channel bitmask for application voice 1, the second is that for application voice 2, and so on, up to four voices.

-f *frag_size*

The requested fragment size.

-m *mixer_name*

The string name for mixer input.

-n *num_frags*

The requested number of fragments.

-p *volume*

The volume, as a percentage.

-v

Be verbose.

Description:

The wave utility plays back audio data.

Copyright © 2014, QNX Software Systems Limited

waverec

Record audio data

Syntax:

```
waverec [options]
```

Runs on:

QNX Neutrino

Options:

-8

Use 8-bit mode instead of the default 16-bit mode.

-a [*card_num*:]*dev_num*

The card and device number to record from.

-b *size*

The sample size; one of 8, 16, or 32.

-c *hw_channel_bitmask*[,*hw_channel_bitmask*...]

Voice matrix configuration. The first argument is the hardware channel bitmask for application voice 1, the second is that for application voice 2, and so on, up to four voices.

-f *frag_size*

The requested fragment size.

-m

Record in mono instead of in stereo.

-n *num_voices*

The number of voices to record. The default is 2 (stereo).

-r *rate*

The rate at which to record; one of 48000, 44100 (the default), 22050, or 11025.

-t *sec*

Copyright © 2014, QNX Software Systems Limited

The number of seconds to record (5 seconds default).

-v

Be verbose.

If you specify both the -m and -n options, the one specified later is used.

Description:

The `waverec` utility captures (i.e., records) audio data.

Copyright © 2014, QNX Software Systems Limited

WC

Count words, lines, and bytes (POSIX)

Syntax:

```
wc [-clw] [-ht] [file...]
```

Runs on:

QNX Neutrino

Options:

-c

Write to the standard output the number of bytes in each input file.

-h

(QNX Neutrino extension) Display headers.

-l

("el") Write to the standard output the number of lines in each input file.

-t

(QNX Neutrino extension) Don't display the totals when counting more than one file.

-w

Write to the standard output the number of words in each input file.

file

The pathname of an input file. If no files are specified, the standard input is used.

Description:

For each input file, the wc utility counts lines, words, and characters (bytes) in the file and writes the results to the standard output. It considers a *word* to be a string of characters delimited by white space.

If you don't specify any options, wc writes counts of lines, words and characters, in that order. If you specify one or more of the options -l, -w, or -c, the wc utility reports only the information selected for the specified options. The order in which you specify

Copyright © 2014, QNX Software Systems Limited

these options determines the order in which the number of lines, words, and bytes are written.

If you specify more than one input file, wc also writes a total count for all named files, for each option. You can specify the -t option to suppress the totals.

To get its line count, wc counts newline characters. If the last line of a file lacks the newline terminator, it isn't counted.

Exit status:

0

Success.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

which

Locate a program file (UNIX)

Syntax:

```
which [-afLls] program...
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-a

Find all occurrences of *program* in ***PATH***.

-f

Display the full pathname.

-L

Display the long format (as in *ls -l* (p. 1139)) for each program found, displaying link information if the file is a symlink.

-l

("el") Display the long format (as in *ls -l* (p. 1139)) for each program found.

-s

Search for shared objects in the directories identified by the ***LD_LIBRARY_PATH*** environment variable and the _CS_LIBPATH configuration string.

Description:

The `which` utility searches for the specified *programs*. By default, `which` searches the directories listed by your ***PATH*** environment variable, but if you specify the -s option, it searches the directories specified by ***LD_LIBRARY_PATH*** and _CS_LIBPATH.

The Korn shell has a builtin *whence* (p. 1082) command that's similar to the `which` utility but also tells you if the given command is a reserved word, alias, builtin command, and so on. For more information, see the entry for `ksh`.

Copyright © 2014, QNX Software Systems Limited

Examples:

Display the full pathname and long status for all versions of the *ls* (p. 1139) utility found in ***PATH***:

```
which -alf ls
```

Display the pathname for the `which` utility:

```
which which
```

Environment variables:

PATH

A colon-separated list of directories to search for executables.

LD_LIBRARY_PATH

A colon-separated list of directories to search for shared libraries.

Exit status:

0

All input files were found.

>0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

wiconfig

Configure WaveLAN/IEEE devices

Syntax:

```
wiconfig interface [-Dho] [-A 1|2] [-a access_point_density]
         [-d max_data_length] [-g fragmentation_threshold] [-M
 0|1]
         [-m MAC_address] [-R 1|3] [-r RTS_threshold] [-s
station_name]
```

Runs on:

QNX Neutrino

Options:

-A 1|2

Set the authentication type for a specified interface. Permitted values are 1 (Open System Authentication) or 2 (Shared Key Authentication). The default is 1.

-a *access_point_density*

Specify the access point density for a given interface. Legal values are 1 (low), 2 (medium), and 3 (high). This setting influences some of the radio modem threshold settings.

-D

This forces the driver to initiate one round of access point scanning. All of the access points found are displayed.

-d *max_data_length*

Set the maximum receive and transmit frame size for a specified interface. The maximum data length can be any number from 256 to 2346. The default is 2304.

-g *fragmentation_threshold*

Set the fragmentation threshold.

-h

Display a short help message.

-M 0|1

Copyright © 2014, QNX Software Systems Limited

Enable or disable “microwave oven robustness” on a given interface. This should only be used if needed.

In cases of slow performance where there is a good quality signal but also high levels of noise (i.e., the signal-to-noise ratio is bad, but the signal strength is good), and there is an operating microwave oven in or near the signal path, this option may be of use.

In bad signal-to-noise conditions, the link layer will switch to lower transmit rates. However at lower transmit rates, individual frames take longer to transmit, making them more vulnerable to bursty noise. The option works by enabling data fragmentation in the link layer as the transmit speed lowers in an attempt to shorten the transmit time of each frame so that individual frames are more likely to be transmitted without error.

Note that this doesn't impact the visible MTU of the link.

-m ***MAC_address***

Set the station address for the specified interface. The MAC address is specified as a series of six hexadecimal values separated by colons, e.g., 00:60:1d:12:34:56. This programs the new address into the card and updates the interface as well.

-o

Print out the statistics counters instead of the card settings. Note that, however, the statistics are updated only every minute or so.

-R 1|3

Enable or disable roaming function on a given interface. The legal values are 1 (roaming handled by firmware) and 3 (roaming disabled). The default is 1.

-r ***RTS_threshold***

Set the RTS/CTS threshold for a given interface. This controls the number of bytes used for the RTS/CTS handshake boundary. The *RTS_threshold* can be any value between 0 and 2347. The default is 2347, which indicates that the RTS/CTS mechanism is never to be used.

-s ***station_name***

Sets the station name, which is used for diagnostic purposes, for the specified interface. The Lucent WaveMANAGER software can poll the names of remote hosts.

Copyright © 2014, QNX Software Systems Limited

Description:

The `wiconfig` command controls the operation of WaveLAN/IEEE wireless networking devices via the `wi` and `awi` drivers. The `wiconfig` command can also be used to view the current settings of these parameters and to dump out the values of the card's statistics counters.

You aren't likely to need this utility; *`ifconfig`* (p. 957) can handle the device configuration required without needing this utility.

With no extra options, `wiconfig` displays the current settings of the specified WaveLAN/IEEE interface.

Most of the parameters that can be changed relate to the IEEE 802.11 protocol which the WaveLAN implements. This includes the station name, whether the station is operating in ad-hoc (point to point) or BSS (service set) mode, and the network name of a service set to join (IBSS) if BSS mode is enabled.

The interface argument given to `wiconfig` should be the logical interface name associated with the WaveLAN/IEEE device (e.g., `wi0`, `wi1`, etc.).

Copyright © 2014, QNX Software Systems Limited

wlanctl

Examine the IEEE 802.11 wireless LAN client/peer table

Syntax:

```
wlanctl interface [...]
wlanctl -a
```

Runs on:

QNX Neutrino

Options:

-a

Display the nodes for all interfaces.

Description:

Use the `wlanctl` utility to print node tables from IEEE 802.11 interfaces. Use the -a flag to print the nodes for all interfaces, or list one or more 802.11 interfaces to select their tables for examination. For example, to examine the node tables for `atw0`, use:

```
wlanctl atw0
```

The `wlanctl` utility displays the node table. For example:

```
atw0: mac 00:02:6f:20:f6:2e bss 02:02:6f:20:f6:2e
        node flags 0001<bss>
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0022<ibss,short preamble>
        beacon-interval 100 TU tsft 18425852102545544165 us
        rates [1.0] 2.0 5.5 11.0
        assoc-id 0 assoc-failed 0 inactivity 0s
        rssi 161 txseq 10 rxseq 1420
atw0: mac 00:02:2d:2e:3c:f4 bss 02:02:6f:20:f6:2e
        node flags 0000
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0002<ibss>
        beacon-interval 100 TU tsft 18425852105450086784 us
        rates [1.0] 2.0 5.5 11.0
        assoc-id 0 assoc-failed 0 inactivity 0s
        rssi 159 txseq 2 rxseq 551
atw0: mac 00:02:6f:20:f6:2e bss 02:02:6f:20:f6:2e
        node flags 0000
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0022<ibss,short preamble>
        beacon-interval 100 TU tsft 18425852102558548069 us
        rates [1.0] 2.0 5.5 6.0 9.0 11.0 12.0 18.0 24.0 36.0 48.0 54.0
        assoc-id 0 assoc-failed 0 inactivity 145s
        rssi 163 txseq 9 rxseq 2563
```

Copyright © 2014, QNX Software Systems Limited

This example is taken from a network consisting of three stations running in ad hoc mode. The key for interpreting the node printouts follows:

mac

In the sample node table, the first network node has MAC number `00:02:6f:20:f6:2e`.

bss

The first node belongs to the 802.11 network identified by Basic Service Set Identifier (BSSID) `02:02:6f:20:f6:2e`.

node flags

Only one node flag, `bss`, is presently defined. The first node is distinguished from the rest by its node flags: flag `bss` indicates that the node represents the 802.11 network that the interface has joined or created. The MAC number for the node is the same as the MAC number for the interface.

ess

The name of the (Extended) Service Set we have joined. This is the same as the network name set by *ifconfig* (p. 957) with the `ssid` option.

chan

The `wlanctl` utility prints the channel number, the center frequency in megahertz, and the channel flags. The channel flags indicate the frequency band ((`2.4GHz` or `5GHz`), modulation (`cck`, `gfsk`, `ofdm`, `turbo`, and `dynamic cck-ofdm`), and operation constraints (`passive scan`). Common combinations of band and modulation are:

Band	Modulation	Description
2.4GHz	cck	11Mb/s DSSS 802.11b
2.4GHz	gfsk	1-2Mb/s FHSS 802.11
2.4GHz	ofdm	54Mb/s 802.11g
2.4GHz	dynamic cck-ofdm	Mixed 802.11b/g network
5GHz	ofdm	54Mb/s 802.11a
5GHz	turbo	108Mb/s 802.11a

capabilities

Copyright © 2014, QNX Software Systems Limited

Ad hoc-mode and AP-mode 802.11 stations advertise their capabilities in 802.11 Beacons and Probe Responses. The `wlanctl` utility understands these capability flags:

Flag	Description
`ess`	Infrastructure (access point) network
`ibss`	Ad hoc network (no access point)
`privacy`	WEP encryption
`short preamble`	Reduce 802.11b overhead
`pbcc`	22Mbps 802.11b+
`channel agility`	Change channel for licensed services

`beacon-interval`

In the example, beacons are sent once every 100 Time Units. A Time Unit (TU) is 1024 microseconds (a “kilo-microsecond” or `kus`). Thus 100 TU is about one tenth of a second.

`tsft`

802.11 stations keep a Time Synchronization Function Timer (TSFT) which counts up in microseconds. Ad hoc-mode stations synchronize time with their peers. Infrastructure-mode stations synchronize time with their access point. Power-saving stations wake and sleep at intervals measured by the TSF Timer. The TSF Timer has a role in the coalescence of 802.11 ad hoc networks (“IBSS merges”).

`rates`

802.11 stations indicate the bit-rates they support, in units of 100KB/s in 802.11 Beacons, Probe Responses, and Association Requests. The `wlanctl` utility prints a station's supported bit-rates in 1Mb/s units. A station's basic rates are flagged by an asterisk (`*`). The last bit-rate at which a packet was sent to the station is enclosed by square brackets.

`assoc-id`

In an infrastructure network, the access point assigns to each client an Association Identifier, which is used to indicate traffic for power-saving stations.

`assoc-failed`

Copyright © 2014, QNX Software Systems Limited

The number of times the station tried and failed to associate with its access point.

inactivity

The number of seconds that elapsed since a packet was last received from the station. When this value reaches `net.link.ieee80211.maxinact`, the station is eligible to be purged from the node table.

rssi

Unitless Received Signal Strength Indication (RSSI). Higher numbers indicate stronger signals. Zero is the lowest possible RSSI. On a hostap- or adhoc-mode interface, the node with node flag `bss` set uses `rssi` to indicate the signal strength for the last packet received from a station that doesn't belong to the network. On an infrastructure-mode station, the node with node flag `bss` set indicates the strength of packets from the access point.

txseq

The next 802.11 packet sent to this station will carry this transmit sequence number. The 802.11 MAC uses the transmit sequence number to detect duplicate packets.

rxseq

The last packet received from this station carried this transmit sequence number.

Copyright © 2014, QNX Software Systems Limited

wpa_cli

WPA command-line client for interacting with `wpa_supplicant`

Syntax:

```
wpa_cli  [-p path to ctrl sockets]
         [-i ifname]
         [-hvB] [-a action file]
         [-P pid file] [command ... ]
```

Runs on:

QNX Neutrino

Options:

-p ***path***

Change the path where control sockets should be found.

-i ***ifname***

Specify the interface that is being configured. By default, choose the first interface found with a control socket in the socket path.

-h

Help. Show a usage message.

-v

Show version information.

-B

Run as a daemon in the background.

-a ***file***

Run in daemon mode executing the action file based on events from `wpa_supplicant`. The specified file will be executed with the first argument set to the interface name, and the second to `CONNECT` or `DISCONNECT`, depending on the event.

-P ***file***

Set the location of the PID file.

command

Copyright © 2014, QNX Software Systems Limited

Run a command; see "*Supported commands* (p. 2060)," below.

Description:

The `wpa_cli` utility is a text-based front-end program for interacting with *`wpa_supplicant`* (p. 2065). You can use it to query the current status, change the configuration, trigger events, and request interactive user input.

The `wpa_cli` utility can show the current authentication status, selected security mode, dot11 and dot1x MIBs, etc. In addition, it can configure some variables like EAPOL state machine parameters and trigger events like reassociation and IEEE 802.1X logoff/logon.

The `wpa_cli` utility provides a user interface to request authentication information, such as user name and password, if these aren't included in the configuration. You can use this to implement, for example, one-time passwords or generic token card authentication where the authentication is based on a challenge-response that uses an external device for generating the response.

You can configure the control interface of `wpa_supplicant` to allow non-`root` user access (`ctrl_interface_group` in the configuration file). This makes it possible to run `wpa_cli` with a normal user account.

The `wpa_cli` utilities supports interactive and command-line modes. Both modes share the same command set, and the main difference is in interactive mode providing access to unsolicited messages (event messages, user name/password requests).

If you don't specify a command when you start `wpa_cli`, the utility goes into interactive mode. You then enter commands at the `wpa_cli` prompt.

Supported commands

The `wpa_cli` utility currently supports the following commands:

`add_network`

Add a network.

`bssid` ***network_id BSSID***

Set the preferred BSSID for an SSID.

`disable_network` ***network_id***

disable a network

`disconnect`

Disconnect and wait for a `reassociate` command before connecting.

`enable_network` ***network_id***

Copyright © 2014, QNX Software Systems Limited

Enable a network.

get_capability *eap/pairwise/group/key_mgmt/proto/auth_alg*

Get capabilities.

get_network *network_id variable*

Get network variables.

help

Display usage information.

identity *network_id identity*

Configure the identity for an SSID.

interface [*ifname*]

Show interfaces or select the specified interface.

level *debug_level*

Change the debugging level.

license

Show the full `wpa_cli` license.

list_networks

List the configured networks.

logoff

IEEE 802.1X EAPOL state machine logoff.

logon

IEEE 802.1X EAPOL state machine logon.

mib

Get MIB variables (`dot1x`, `dot11`)

new_password *network_id password*

Change the password for an SSID.

otp *network_id password*

Configure a one-time password for an SSID.

passphrase *network_id passphrase*

Copyright © 2014, QNX Software Systems Limited

Configure a private key passphrase for an SSID.

password *network_id password*

Configure a password for an SSID.

pin *network_id pin*

Configure a pin for an SSID.

pmksa

Show the PMKSA cache.

preauthenticate *BSSID*

Force preauthentication.

quit

Exit `wpa_cli`

reassociate

Force a reassociation.

reconfigure

Force `wpa_supplicant` to reread its configuration file.

remove_network *network_id*

Remove a network.

save_config

Save the current configuration.

scan

Request a new BSS scan.

scan_results

Get the latest scan results.

select_network *network_id*

Select a network (disable others).

set

Set variables (shows list of variables when run without arguments).

set_network *network_id variable value*

Copyright © 2014, QNX Software Systems Limited

Set network variables (shows list of variables when run without arguments).

status [verbose]

Get the current WPA/EAPOL/EAP status.

terminate

Terminate `wpa_supplicant`.

Copyright © 2014, QNX Software Systems Limited

wpa_passphrase

Set WPA passphrase for a SSID

Syntax:

```
wpa_passphrase [ssid] [passphrase]
```

Runs on:

QNX Neutrino

Options:

ssid

The SSID whose passphrase should be derived.

passphrase

Use this *passphrase*. If not included on the command line, it will be read from standard input.

Description:

The `wpa_passphrase` utility precomputes PSK entries for network configuration blocks of a `wpa_supplicant.conf` file.

Copyright © 2014, QNX Software Systems Limited

wpa_supplicant

Wi-Fi Protected Access client and IEEE 802.1X supplicant

Syntax:

```
wpa_supplicant [-BhKLNptuvW] [-b br_ifname]
               [-C ctrl_interface] [-c config file] [-d[d]]
               [-f output_file] [-g global ctrl_interface]
               [-i ifname] [-P file] [-q[q]]
```

Runs on:

QNX Neutrino

Options:

Most command-line options have global scope. Some are given per interface, and are valid only if you've specified at least one -i option; otherwise they're ignored. Option groups for different interfaces must be separated by an -N option.

-B

Run as a daemon in the background.

-b *br_ifname*

Optional bridge interface name. (Per interface)

-C *ctrl_interface*

The path to the *ctrl_interface* socket. (Per interface; used only if -c is not).

-c *filename*

Path to configuration file. (Per interface)

-d

Increase debugging verbosity (specify -dd for even more).

-f*output_file*

Send the output to the specified file, instead of to standard output.

-g *global ctrl_interface*

The path to the *global ctrl_interface* socket. If you specify this option, you can omit the interface definitions.

-h

Copyright © 2014, QNX Software Systems Limited

Help; display a usage message.

-i ***ifname***

The interface to listen on. Multiple instances of this option can be present, one per interface, separated by an -N option (see below).

-K

Include keys (passwords, etc.) in the debugging output.

-L

Show the license (GPL and BSD).

-N

Start describing a new interface.

-P ***file***

Specify the location of the PID file.

-p

Driver parameters. (Per interface)

-q

Decrease debugging verbosity (specify -qq for even less).

-t

Include the timestamp in debugging messages.

-u

Enable the DBus control interface. If you specify this option, you can omit the interface definitions.

-v

Show version information.

-W

Wait for a control interface before starting.

Description:

Wireless networks don't require physical access to the network equipment in the same way as wired networks. This makes it easier for unauthorized users to passively monitor a network and capture all transmitted frames. In addition, unauthorized use of the network is much easier. In many cases, this can happen even without the user's explicit

Copyright © 2014, QNX Software Systems Limited

knowledge, because the wireless LAN adapter may have been configured to automatically join any available network.

Link-layer encryption can be used to provide a layer of security for wireless networks. The original wireless LAN standard, IEEE 802.11, included a simple encryption mechanism, WEP. However, that proved to be flawed in many areas, and networks protected with WEP cannot be considered to be secure.

IEEE 802.1X authentication and frequently-changed dynamic WEP keys can be used to improve the network security, but even that has inherited security issues, due to the use of WEP for encryption. Wi-Fi Protected Access and IEEE 802.11i amendment to the wireless LAN standard introduce a much-improved mechanism for securing wireless networks. IEEE 802.11i-enabled networks that are using CCMP (an encryption mechanism based on strong cryptographic algorithm AES) can finally be called secure used for applications which require efficient protection against unauthorized access.

The `wpa_supplicant` utility is an implementation of the WPA Supplicant component, i.e., the part that runs in the client stations. It implements WPA key negotiation with a WPA Authenticator and EAP authentication with Authentication Server. In addition, it controls the roaming and IEEE 802.11 authentication/association of the wireless LAN driver.

The `wpa_supplicant` utility is designed to be a daemon that runs in the background and acts as the backend component controlling the wireless connection. It supports separate front-end programs and a sample text-based front-end, *`wpa_cli`* (p. 2059), is included with `wpa_supplicant`.

Before `wpa_supplicant` can do its work, the network interface must be available. That means that the physical device must be present and enabled, and the driver for the device must have been loaded. The daemon exits immediately if the device isn't already available.

After `wpa_supplicant` has configured the network device, higher-level configuration such as DHCP may proceed. There's a variety of ways to integrate `wpa_supplicant` into a machine's networking scripts, a few of which are described in the sections below.

The following steps are used when associating with an AP using WPA:

1. `wpa_supplicant` requests the driver to scan neighboring BSSs.
2. `wpa_supplicant` selects a BSS based on its configuration.
3. `wpa_supplicant` requests the driver to associate with the chosen BSS.
4. If WPA-EAP: integrated IEEE 802.1X Supplicant or external Xsupplicant completes EAP authentication with the authentication server (proxied by the Authenticator in the AP).
5. If WPA-EAP: the master key is received from the IEEE 802.1X Supplicant.
6. If WPA-PSK: `wpa_supplicant` uses PSK as the master session key.

Copyright © 2014, QNX Software Systems Limited

7. `wpa_supplicant` completes WPA 4-Way Handshake and Group Key Handshake with the Authenticator (AP).
8. `wpa_supplicant` configures encryption keys for unicast and broadcast.
9. Normal data packets can be transmitted and received.

Supported features

Supported WPA/IEEE 802.11i features:

- WPA-PSK ("WPA-Personal")
- WPA with EAP (e.g., with RADIUS authentication server) ("WPA-Enterprise") The following authentication methods are supported with an integrated IEEE 802.1X Supplicant:
  - EAP-TLS
  - EAP-PEAP/MSCHAPv2 (both PEAPv0 and PEAPv1)
  - EAP-PEAP/TLS (both PEAPv0 and PEAPv1)
  - EAP-PEAP/GTC (both PEAPv0 and PEAPv1)
  - EAP-PEAP/OTP (both PEAPv0 and PEAPv1)
  - EAP-PEAP/MD5-Challenge (both PEAPv0 and PEAPv1)
  - EAP-TTLS/EAP-MD5-Challenge
  - EAP-TTLS/EAP-GTC
  - EAP-TTLS/EAP-OTP
  - EAP-TTLS/EAP-MSCHAPv2
  - EAP-TTLS/EAP-TLS
  - EAP-TTLS/MSCHAPv2
  - EAP-TTLS/MSCHAP
  - EAP-TTLS/PAP
  - EAP-TTLS/CHAP
  - EAP-SIM
  - EAP-AKA
  - EAP-PSK
  - EAP-PAX
  - LEAP (note: requires special support from the driver for IEEE 802.11 authentication)

  The following methods are supported, but since they don't generate keying material, they can't be used with WPA or IEEE 802.1X WEP keying:

  - EAP-MD5-Challenge
  - EAP-MSCHAPv2
  - EAP-GTC
  - EAP-OTP
- key management for CCMP, TKIP, WEP104, WEP40

Copyright © 2014, QNX Software Systems Limited

- RSN/WPA2 (IEEE 802.11i)
  - pre-authentication
  - PMKSA caching

Files:

The `wpa_supplicant` requires the following libraries and binaries be present:

- `libcrypto.so` — crypto library
- `libssl.so` — Secure Socket Library (created from OpenSSL)
- *`random`* (p. 1654) — executable that creates `/dev/urandom` for random-number generation
- `libm.so` — math library required by `random`
- `libz.so` — compression library required by `random`

The `wpa_supplicant` also needs a read/write filesystem for creation of a `ctrl_interface` directory (see the sample `wpa_supplicant.conf` configuration file).

You can't use `/dev/shmem` because it isn't possible to create a directory there.

Copyright © 2014, QNX Software Systems Limited

Chapter 25
X

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “X”.

Copyright © 2014, QNX Software Systems Limited

xargs

Construct argument list(s) and invoke a program (POSIX)

Syntax:

```
xargs [-itx] [-n numargs] [-P n] [-s size]
      [program [initial-arguments]]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

-i

(QNX Neutrino extension) Execute in "insert mode." The *program* is executed *once* for each item in standard input. Each occurrence of `{}` in *initial-arguments* is replaced with the argument read from standard input. If there are no occurrences of `{}` in *initial-arguments*, the argument is appended to the initial list.

-n *numargs*

The maximum number of arguments to append to the command line. The default for *numargs* is 255.

-P *n*

(QNX Neutrino extension) Use up to *n* concurrent commands. The default is 1.

-s *size*

Set the maximum command buffer to *size* characters, including *program* and *initial-argument*. The default for *size* is 4096.

-t

Trace; print each *program* on standard error before executing.

program

The name of the program to execute. The program must be found by searching the path using the ***PATH*** environment variable. If you don't specify *program*, the default is the *echo* (p. 668) utility.

Copyright © 2014, QNX Software Systems Limited

initial-arguments

One or more arguments to *program* that are presented every time *program* is executed.

-x

Terminate if the command line is too long when using *numargs* (or the default number of) arguments.

Description:

The `xargs` utility uses character strings, read from standard input, to construct a command line which it executes. The specified *program* and *initial-arguments* are placed at the beginning of the command line, followed by some number of character strings read. This process continues until the end of the file.

The utility executes a given program with *initial-arguments* one or more times using the parameters read from standard input. The number of strings appended may be limited by the -n option; the size of the command line may be limited by the -s option.

The strings are separated by blanks or newlines, which may be embedded in the strings by prefixing them with `\` or enclosing them in quotes (`"`). To use the quote character as itself, you must prefix it with a `\`.

The -i option causes the command to be executed for each string read. Instead of the normal process of appending the string to the command buffer, the *initial-arguments* are scanned, and every occurrence of `{}` is replaced by the string. If `{}` doesn't occur in *initial-arguments*, the string is appended to the command line and executed.

When a program is executed, it inherits standard output and standard error from `xargs`. Standard input is set to the controlling tty.

The `xargs` utility always limits the total command buffer size to 4096 characters. The following example may be used to verify the integrity of data files on a floppy disk (mounted as `/fd`):

```
find /fd -print | xargs cksum | diff check_file -
```

In the example above, *find* (p. 750) prints the name of each file on the mounted filesystem. The `xargs` utility groups the filenames up for *cksum* (p. 131) to minimize the number of times `cksum` must be executed. The *diff* (p. 619) utility is then used to verify that the calculated checksums are the same as recorded in the `check_file` file.

It's important to note that the following command:

```
find /fd -exec cksum {} \; | diff check_file -
```

Copyright © 2014, QNX Software Systems Limited

achieves the same thing, but requires *cksum* (p. 131) to be reloaded once for each file in `/fd`.

Examples:

Use *cmp* (p. 134) to determine whether the files in the directory `old_data` are the same as the files in the directory `new_data`:

```
ls old_data | xargs -i cmp old_data/{} new_data/{}
```

Display the files in the current working directory, and all subdirectories, in two columns:

```
find . -print | xargs -n 2 echo
```

Exit status:

0

All invocations of *program* completed successfully.

> 0

An error occurred.

Copyright © 2014, QNX Software Systems Limited

Chapter 26
Y

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

There are currently no “Y” entries.

Copyright © 2014, QNX Software Systems Limited

Chapter 27
Z

The QNX Neutrino resource managers and utilities are described here in alphabetical order.

A ¦ *B* ¦ *C* ¦ *D* ¦ *E* ¦ *F* ¦ *G* ¦ *H* ¦ *I* ¦ *J* ¦ *K* ¦ *L* ¦ *M* ¦ *N* ¦ *O* ¦ *P* ¦ *Q* ¦ *R* ¦ *S* ¦ *T* ¦ *U* ¦ *V* ¦ *W* ¦ *X* ¦ *Y* ¦ *Z*

This chapter describes the utilities, etc. whose names start with “Z”.

Copyright © 2014, QNX Software Systems Limited

zap

Destroy a damaged file (QNX)

Syntax:

```
zap [-pv] file
zap [-pv] [-l|-u] directory
```

Runs on:

QNX Neutrino

Options:

-l

("el") List previously zapped files in the *directory*.

-p

Pause before starting (for floppy disks).

-u

"Unzap" files in the *directory*.

-v

Be verbose; report results.

file

The name of the file to remove.

directory

The name of the directory containing previously zapped files.

Description:

You should use zap to remove a file if:

- you know that the file contains bad disk blocks

  Or:

- *chkfsys* (p. 114) has instructed you to use zap on it.

The zap utility releases a file by clearing the directory entry for that file. The disk blocks used by the file aren't reclaimed. Therefore, if you use zap repeatedly, you'll

Copyright © 2014, QNX Software Systems Limited

reduce the total number of disk blocks available on the disk. You can reclaim these, however, by running `chkfsys` when the system is idle.

Normally, you should use the *rm* (p. 1673) or *rmdir* (p. 1675) commands to release files or directories.

Previously zapped files may be listed in any specified directory using the -l option.

You can "unzap" or recover zapped files by using the -u option. The utility prompts for each file that was zapped in the specified directory.

The file to be "unzapped" *must* have been initially removed via a `zap` command. Files removed conventionally via *rm* (p. 1673) or any other process which calls *unlink()* can't be restored by means of the `zap -u` command.

Examples:

Eliminate the directory entry for the file `junk`:

```
zap junk
```

Exit status:

0

Success.

>0

An error occurred.

Caveats:

To run `zap` you must have read and write permission for the block special file for the filesystem containing the file being zapped. You must also have execute permission for the `zap` utility. In a normally configured system this means you must be `root` to run `zap`.

Copyright © 2014, QNX Software Systems Limited

zcat

Concatenate compressed files (UNIX)

Syntax:

```
zcat [-hLV] [name...]
```

Runs on:

QNX Neutrino, Microsoft Windows

Options:

See *gzip* (p. 921) for a complete listing.

Description:

The `zcat` utility is part of the `gzip` suite. See `gzip` for a description.

This utility is subject to the GNU Public License (GPL). We've included it for use on development systems.

Contributing author:

GNU

Copyright © 2014, QNX Software Systems Limited

zip

Archive and package files to a pkzip format

Syntax:

```
zip [-options] [-b path] [-t mmddyyyy] [-n suffixes]
    [zipfile list] [-xi list]
```

Runs on:

QNX Neutrino, Microsoft Windows

Targets:

x86 only

Options:

-0 to -9

Regulate the speed of compression, where -0 indicates no compression, -1 (“one”) indicates the fastest compression method (less compression) and -9 indicates the slowest compression method (optimal compression). The default compression level is -6.

-@

Read names from standard input.

-A

Adjust self-extracting `exe`.

-b *path*

Use the specified *path* for the temporary `zip` archive.

-c

Add one-line comments for each file. File operations (adding, updating) are done first, and then the utility prompts for one-line comments for each file.

-D

Don't create entries in the zip file for directories. Directory entries are created by default to allow their attributes to be saved in the zip archive.

-d

Copyright © 2014, QNX Software Systems Limited

Delete entries from a zip file.

-e

Encrypt the archive.

-F

Fix the `zip` file. Use this option if some portions of the file are missing.

-FF

Try harder to fix the `zip` file.

-f

Freshen (replace) only the files that have changed.

-h

Display a help message.

-h2

Show more help.

-i

Include only specified files. For example:

```
zip -r foo . -i \*.c
```

This command includes only the files that end in `.c` in the current directory and its subdirectories.

-J

Strip any prepended data (i.e. a SFX stub) from the file.

-j

Junk the path to the file; store just the name of the saved file.

-L

Display the `zip` license.

-l

Convert the UNIX end-of-line character LF to the MS-DOS convention, CR LF. Use -ll to convert the CR LF convention back to the end-of-line character, LF.

Don't use these options on binary files.

Copyright © 2014, QNX Software Systems Limited

-m

Move the specified files into the `zip` file and delete the target directories/files.

-n ***suffixes***

Don't attempt to compress files named with the suffixes indicated. You can use either colons or semicolons to separate the suffixes.

-o

Set the "last modified" time on the `zip` file to match the "last modified" time on the `zip` archive entries.

-P ***password***

Use the given password to encrypt the archive entries.

Specifying a plain-text password on the command line or in a script can be a security problem.

-q

Change to quiet mode by eliminating informational messages and command prompts.

-R

Recurse into the directories starting at the current directory.

-r

Recurse into the directories.

-T

Test the integrity of the new `zip` file.

-t ***mmddyyyy***

Don't operate on files modified prior to the specified date, where *mm* is the month, *dd* represents the day of the month, and *yyyy* is the year.

-tt ***mmddyyyy***

Don't operate on files modified after or at the specified date, where *mm* is the month, *dd* represents the day of the month, and *yyyy* is the year.

-u

Update only changed or new files.

Copyright © 2014, QNX Software Systems Limited

-v

Be verbose, or print version information.

-X

Don't save extra file attributes.

-x *files*

Explicitly exclude the files specified. For example:

```
zip -r foo foo -x \*.o
```

includes the contents of `foo` in `foo.zip`, while it excludes all the files that end in `.o`.

-y

Store symbolic links as links instead of as the referenced files.

-z

Add a comment to the zip file.

Description:

The `zip` utility is a compression and file-packaging utility. A companion program (*`unzip`* (p. 2019)), unpacks `zip` archives. The `zip` and `unzip` programs can work with archives produced by PKZIP; PKZIP and PKUNZIP can work with archives produced by the `zip` utility.

The `zip` utility is useful for packaging a set of files for distribution, for archiving files, and for saving disk space by temporarily compressing unused files or directories.

This utility puts one or more compressed files into a single `zip` archive, along with information about the files, such as name, path, date, time of last modification, protection, and check information to verify file integrity.

You can use a single command to pack an entire directory structure into a `zip` archive. Compression ratios of 2:1 to 3:1 are common for text files. The `zip` utility has one compression method (deflation) and can also store files without compression; `zip` automatically chooses the better of the two for each file to be compressed.

When `zip` is given the name of an existing `zip` archive, it replaces identically named entries in the archive or adds entries for new names.

For example, if `foo.zip` exists and contains `foo/file1` and `foo/file2`, and the directory `foo` contains the files `foo/file1` and `foo/file3`, then:

```
zip -r foo foo
```


Copyright © 2014, QNX Software Systems Limited

replaces `foo/file1` in `foo.zip` and adds `foo/file3` to `foo.zip`. Then, `foo.zip` contains `foo/file1`, `foo/file2`, and `foo/file3`; `foo/file2` remains unchanged.

Examples:

Create an archive called `stuff.zip`, for example, and put all the files in the current directory in it, in compressed form:

```
zip stuff *
```

The `.zip` suffix is added automatically, unless the archive name given contains a dot already; this allows the explicit specification of other suffixes.

Because of filename substitution, files starting with "." are not included; to include these to the file:

```
zip stuff .* *
```

This command doesn't include any subdirectories from the current directory.

Zip up an entire directory:

```
zip -r foo foo
```

This command creates the archive `foo.zip`, containing all the files and directories in the directory `foo` that are contained within the current directory.

You may want to make a `zip` archive that contains the files in `foo`, without recording the directory name `foo`. Use the -j option to leave off the paths:

```
zip -j foo foo/*
```

If you're short on disk space, you might not have enough room to hold both the original directory and the corresponding compressed zip archive. In this case, you can create the archive in steps using the -m option.

For example, if `foo` contains the subdirectories `tom`, `dick`, and `harry`, you can perform these commands:

```
zip -rm foo foo/tom
zip -rm foo foo/dick
zip -rm foo foo/harry
```

to create a directory called `foo.zip`. The first command creates the `foo.zip` directory, and the next two commands add to it. As each `zip` command completes, the last created archive is deleted, making room for the next `zip` command to function.

Environment variables:

ZIPOPT

Copyright © 2014, QNX Software Systems Limited

A set of default options for `zip`. For example:

```
export ZIPOPT="-D"
```

Exit status:

0

The operation succeeded.

2

An error occurred; the operation failed.

3

A generic error in the zipfile format was detected, but processing may have completed successfully anyway; a warning was generated in the process.

4

`zip` was unable to allocate memory for one or more buffers during program initialization.

5

A severe error in the zipfile format was detected; processing probably failed immediately.

6

Entry too large to be split with zipsplit.

7

Invalid comment format.

8

`zip` -T failed or out of memory.

9

The user aborted `zip` prematurely with **Ctrl**–**C** or a similar command.

10

`zip` encountered an error while using a temporary file.

11

Read or seek error.

12

Copyright © 2014, QNX Software Systems Limited

`zip` has nothing to do.

13

Missing or empty `zip` file.

14

Error writing to a file.

15

`zip` was unable to create a file to write to.

16

Bad command-line parameters.

18

`zip` could not open a specified file to read.

Copyright © 2014, QNX Software Systems Limited

Appendix A
Commonly Used Environment Variables

This appendix describes most of the environment variables that are commonly used in the QNX Neutrino RTOS.

For general information about setting environment variables, see the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*; for specific information, see the related utilities and programs.

Copyright © 2014, QNX Software Systems Limited

A

AUTOCONNECT

In order for the */etc/autoconnect* (p. 50) to be run, this environment variable must be defined and set to 1.

For more information, see */etc/autoconnect* (p. 50) in the *Utilities Reference*.

Copyright © 2014, QNX Software Systems Limited

B

BROADCAST

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the client broadcast address that was obtained from the server.

Copyright © 2014, QNX Software Systems Limited

C

COLUMNS

The number of character columns on the screen.

Copyright © 2014, QNX Software Systems Limited

D

DISPLAY

The name of the physical display on which to draw.

DL_DEBUG

If this environment variable is set, the shared library loader displays debugging information about the libraries as they're opened. The value can be a comma-separated list of the following:

- `all` — display all debug messages.
- `help` — display a help message, and then exit.
- `reloc` — display relocation processing messages.
- `libs` — display information about shared objects being opened.
- `statistics` — display runtime linker statistics.
- `lazyload` — print lazy-load debug messages.
- `debug` — print various runtime linker debug messages.

A value of 1 (one) is the same as `all`.

DONT_USE_LINK_UNLINK

Indicates to *lpr* (p. 1128) to use *rename()* instead of *link()* or *unlink()*.

Copyright © 2014, QNX Software Systems Limited

E

EDITOR

The path of the editor you'd like to use by default.

EXINIT

Default *elvis* (p. 672) option settings. If set, the contents of this environment variable are executed on startup as a series of ex commands.

Copyright © 2014, QNX Software Systems Limited

F

FILENAME

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the filename supplied in the server response.

Copyright © 2014, QNX Software Systems Limited

G

GATEWAY

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the gateway that the client is to use.

GNUTARGET

Specifies the target (output file format) for GNU utilities. For more information, see the *Selecting the Target System* appendix of the *Utilities Reference*.

GZIP

A set of default options for *gzip* (p. 921)

Copyright © 2014, QNX Software Systems Limited

H

HOME

Your current working directory when you first log in. It's specified as one of the fields for each user in `/etc/passwd`. For more information, see *passwd* (p. 1434) in the *Utilities Reference*.

HOSTALIASES

The name of a file containing aliases for hosts. For more information, see *gethostbyname()* in the *C Library Reference*.

HOSTNAME

The name of the host machine.

Copyright © 2014, QNX Software Systems Limited

I

INTERFACE

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the interface that was configured (e.g. en0).

IOPORT

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the hex address of the I/O port (e.g. 320).

IOPORT2

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the hex address of the second I/O port, if assigned.

IOPORT2SZ

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the size of the second I/O port, if assigned.

IOPORTSZ

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the size of the I/O port (e.g. 32).

IPADDRESS

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the client IP address that was obtained from the server.

IRQ

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the IRQ of the device.

IVE_HOME

Used by Java VM.

Copyright © 2014, QNX Software Systems Limited

J

J9PLUGIN_ARGS

Arguments passed to Browser Java plugins.

Copyright © 2014, QNX Software Systems Limited

L

LANG

The locale to use for the locale categories.

QNX Neutrino currently supports only the POSIX (i.e. C) locale.

LC_TYPE

The locale for character classification, used by *uniq* (p. 2015) to determine the characters constituting a blank in the current locale.

QNX Neutrino currently supports only the POSIX (i.e. C) locale.

LDEMULATION

Specifies the linker emulation. For more information, see the *Selecting the Target System* appendix of the *Utilities Reference*.

LD_BIND_NOW

Affects lazy-load dependencies due to full symbol resolution. Typically, it forces the loading of all lazy-load dependencies (until all symbols have been resolved).

LD_DEBUG

A synonym for ***DL_DEBUG***. If you set both ***DL_DEBUG*** and ***LD_DEBUG***, then ***DL_DEBUG*** takes precedence.

LD_DEBUG_OUTPUT

The name of a file in which the dynamic linker writes its output. By default, output is written to *stderr*.

For security reasons, the use of ***LD_DEBUG_OUTPUT*** with `setuid` binaries is disabled.

LD_LIBRARY_PATH

A path to search for shared libraries for a native linker. For more information, see *ld* (p. 1096) in the *Utilities Reference*

Copyright © 2014, QNX Software Systems Limited

LD_PRELOAD

A list of full paths to the shared libraries on an ELF system that you want to load before loading other libraries. Use a colon (`:`) to separate the libraries in this list. You can use this environment variable to add or change functionality when you run a program. For `setuid` or `setgid` ELF binaries, only libraries in the standard search directories that are also `setuid` will be loaded. See *ld* (p. 1096) in the *Utilities Reference*, and *dlopen()* in the QNX Neutrino *C Library Reference*.

LD_RUN_PATH

A path to search for shared libraries on an ELF system. For more information, see *ld* (p. 1096) in the *Utilities Reference*

LEASEEXPIRES

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the time at which the lease expires.

LEASEOBTAINED

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the time at which the lease was obtained.

LESS

Options that you want to pass to *less* (p. 1100) automatically.

LESSEDIT

The editor prototype string (used for the `v` command in *less* (p. 1100)).

LINES

The number of character lines on the screen.

LOCALDOMAIN

The local domain name. For more information, see *res_init()* in the *C Library Reference*.

LOGNAME

The userid you used to login; the same as ***USERNAME***.

Copyright © 2014, QNX Software Systems Limited

M

MAKEFLAGS

A set of default options for *make* (p. 1166).

MALLOC_OPTIONS

Controls how *calloc()*, *malloc()*, and *realloc()* behave if you specify a size of 0 (or a value of 0 for the *n* argument to *calloc()*). The V ("System V") and R ("use the *realloc()* behavior of QNX Neutrino 6.4.0 and earlier") columns below indicate how the functions behave if the value of ***MALLOC_OPTIONS*** includes that letter:

Function	Default	V	R
calloc(n, 0)	Non-NULL	NULL	No effect
malloc(0)	Non-NULL	NULL	No effect
realloc(NULL, 0)	Non-NULL	NULL	No effect
realloc(non-NULL, 0)	Non-NULL	NULL	NULL

In all the above cases, if the function returns a non-NULL pointer, it's valid only for a corresponding call to *free()* or *realloc()*.

MKIFS_PATH

A colon-separated list of directories that *mkifs* (p. 1241) should use to search for host files to be included in an OS image.

MORE

Default options for *more* (p. 1309)

Copyright © 2014, QNX Software Systems Limited

N

NAMESERVER1*, *NAMESERVER2

The *dhcp.client* (p. 544) passes these environment variables to the `/etc/dhcp/dhcp-up` script. They indicate the nameservers that the client is to use.

NAME_MAX

The maximum permitted length of a component of a pathname.

NETMASK

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the client netmask that was obtained from the server.

Copyright © 2014, QNX Software Systems Limited

O

OPTIONx

The *dhcp.client* (p. 544) passes these additional environment variables (where *x* is the option number) to the `/etc/dhcp/dhcp-up` script.

Copyright © 2014, QNX Software Systems Limited

P

PATH

A colon-separated list of directories that are searched when the shell looks for commands and `.`'d files. An empty string resulting from a leading or trailing colon, or two adjacent colons is treated as a `.`, the current directory.

For more information, see *ksh* (p. 1029) in the *Utilities Reference*.

PATH_MAX

The maximum permitted length of a pathname.

POSIX_STRICT

If this environment variable is set, some utilities (e.g., *cp* (p. 141), *ls* (p. 1139), and *more* (p. 1309)) interpret options according to POSIX specifications.

POSIXLY_CORRECT

This environment variable is used by Unix-style operating systems to alter behavior to comply with POSIX where it's different from the OS's default behavior. ***POSIXLY_CORRECT*** is a *de facto* standard that isn't defined by POSIX.

Here are some of its effects:

- If the ***POSIXLY_CORRECT*** environment variable is set, functions that check the length of a pathname do so *before* removing any redundant `.` and `..` components. If ***POSIXLY_CORRECT*** isn't set, the functions check the length *after* removing any redundant components.
- If the ***POSIXLY_CORRECT*** environment variable is set, the posix option to be enabled for `ksh`. For more information, see "*POSIX mode* (p. 1056)" in the documentation for `ksh`.

PRINTER

The name of the default printer, used by *lpr* (p. 1128).

PROCESSOR

Specifies the target CPU when building an image filesystem. If not set, the default is the same as the CPU of the host system (e.g. `x86`). For more information, see *mkifs* (p. 1241) in the *Utilities Reference*.

PYTHONCASEOK

Copyright © 2014, QNX Software Systems Limited

Ignore case in Python `import` statements (Windows).

PYTHONDEBUG

Display debugging output from the *python* (p. 1603) parser.

PYTHONHOME

An alternate *prefix* directory (or *prefix*:*exec_prefix*) for Python. The default module search path uses *prefix*/`python`*X.X*.

PYTHONINSPECT

Inspect interactively after running the Python script, and force prompts, even if *stdin* doesn't appear to be a terminal.

PYTHONOPTIMIZE

Optimize the generated Python byte code.

PYTHONPATH

A colon-separated list of directories prefixed to the default module search path for Python. The result is stored in *sys.path*.

PYTHONSTARTUP

The file to execute on interactive startup of Python (no default).

PYTHONUNBUFFERED

Use unbuffered binary for *stdout* and *stderr*. See the Python documentation for details on internal buffering.

PYTHONVERBOSE

Make Python be verbose (trace import statements).

Copyright © 2014, QNX Software Systems Limited

Q

QCC_CONF_PATH

The name of the directory that contains the configuration files for *qcc* (p. 1608).

QNX_HOST

The location of host-specific files for all development hosts.

QNX_TARGET

The location of target backends on the host machine.

Copyright © 2014, QNX Software Systems Limited

R

RESCONF

An way of overriding configuration strings in the *resolv.conf* (p. 1663) file.

Copyright © 2014, QNX Software Systems Limited

S

SERVER

The *dhcp.client* (p. 544) passes this environment variable to the `/etc/dhcp/dhcp-up` script. It indicates the server's IP address.

SHELL

The pathname of the command interpreter, or shell, that you want to use. It's set by *login* (p. 1121).

SOCKET

The *pccard-launch* (p. 1451) utility sets this environment variable to indicate the socket where the card is inserted.

SOCKS_NS

The IP address of the domain nameserver that should be used for name resolution by SOCKS client programs.

SOCKS_SERVER

The name or IP address of the SOCKS proxy server host to use, overriding the default server for SOCKS client programs.

STDIO_DEFAULT_BUFSIZE

As a QNX Neutrino extension to POSIX standard I/O, you can use this environment variable to override `BUFSIZ` as the default buffer size for stream I/O. The value of ***STDIO_DEFAULT_BUFSIZE*** must be greater than that of `BUFSIZ`.

SYSLOG

The node on which *syslogd* (p. 1885) is running.

SYSNAME

The name of the system.

Copyright © 2014, QNX Software Systems Limited

T

TERM

The terminal type.

TERMINFO

The name of the directory for terminfo files; see *tic* (p. 1960) in the *Utilities Reference*.

TMPDIR

The name of a directory where utilities can create temporary files.

TZ

The timezone definition; see "Setting the time zone" in the Configuring Your Environment chapter of the QNX Neutrino *User's Guide*.

Copyright © 2014, QNX Software Systems Limited

U

USER

The userid you used to login; the same as ***LOGNAME***.

USER_NAME

Internal use only.

Copyright © 2014, QNX Software Systems Limited

Appendix B
Selecting the Target System

This appendix describes how you can specify targets and architectures for the ELF utilities.

The development tools automatically take care of this — you'll probably never need to worry about it.

You can specify three aspects of the target system to the utilities that work on ELF files, each in several ways:

- target — the object file format
- architecture — the type of CPU the target file is to run on
- linker emulation (which applies to the linker only) — a “personality” that specifies default values for the target system.

In the following summaries, the lists of ways to specify values are in order of decreasing precedence; the ways listed first override those listed later.

Copyright © 2014, QNX Software Systems Limited

Target selection

A *target* is an object file format. A given target may be supported for multiple architectures (see "*Architecture selection*"). A target selection may also have variations for different operating systems or architectures.

The command to list valid target values is `objdump -i` (the first column of output contains the relevant information). A sample value is `elf32-i386`.

Here's how to specify the target for the ELF utilities:

`objdump`

1. -b command-line option
2. ***GNUTARGET*** environment variable
3. Deduced from the input file.

`objcopy`, `strip`

Input target:

1. -I or -F command-line option
2. ***GNUTARGET*** environment variable
3. Deduced from the input file.

Output target:

1. -O or -F command-line option
2. Input target (see above)
3. ***GNUTARGET*** environment variable
4. Deduced from the input file.

`nm`, `size`, and `strings`

1. --target command-line option
2. ***GNUTARGET*** environment variable
3. Deduced from the input file.

`ld`

Input target:

1. -b command-line option
2. TARGET script command (see "Option Commands" in *Using LD* in the full online GNU documentation)

Copyright © 2014, QNX Software Systems Limited

3. ***GNUTARGET*** environment variable
4. Default target of the selected linker emulation (see "*Linker emulation selection*," below)

Output target:

1. --oformat command-line option
2. `OUTPUT_FORMAT` script command (see "Option Commands" in *Using LD* in the full online GNU documentation)
3. Linker input target (see above).

Copyright © 2014, QNX Software Systems Limited

Architecture selection

An *architecture* is a type of CPU on which an object file is to run. Its name may contain a colon, separating the name of the processor family from the name of the particular CPU.

The command to list valid architecture values is `objdump -i`.

Here's how to specify the architecture for the GNU utilities:

`objdump`

1. -m command-line option
2. Deduced from the input file.

`objcopy`, `nm`, `size`, `strings`

Deduced from the input file.

`ld`

Input architecture is deduced from the input file.

Output architecture:

1. `OUTPUT_ARCH` script command (see "Option Commands" in *Using LD* in the full online GNU documentation)
2. Default architecture from the linker output target (see "*Target selection*," above).

Copyright © 2014, QNX Software Systems Limited

Linker emulation selection

A linker *emulation* is a "personality" of the linker; it gives the linker default values for the other aspects of the target system. In particular, it consists of the linker script, the target, and several "hook" functions that are run at certain stages of the linking process to do special things that some targets require.

The command to list valid linker emulation values is `ld -V`. Sample values include `i386nto`.

Ways to specify:

1. -m command-line option
2. ***LDEMULATION*** environment variable
3. Compiled-in `DEFAULT_EMULATION` from `Makefile`, which comes from `EMUL` in `config/`*target*`.mt`.

Copyright © 2014, QNX Software Systems Limited

Appendix C
What's New in this Reference?

This appendix describes the updates to this document in the following releases:

Release	New	Deprecated	Changed	Errata
QNX Neutrino 6.6	*New*	*Deprecated*	*Changed*	*Errata*
QNX SDP 6.5.0 Service Pack 1	*New*		*Changed*	*Errata*
QNX SDP 6.5.0	*New*	*Deprecated*	*Changed*	*Errata*
QNX SDP 6.4.1	*New*	*Deprecated*	*Changed*	*Errata*
QNX SDP 6.4.0	*New*	*Deprecated*	*Changed*	*Errata*
QNX Momentics 6.3.2	*New*		*Changed*	*Errata*
QNX Neutrino Core OS 6.3.2	*New*		*Changed*	*Errata*
QNX Momentics 6.3.0 Service Pack 2	*New*		*Changed*	*Errata*
QNX Momentics 6.3.0 Service Pack 1	*New*		*Changed*	
QNX Momentics 6.3.0	*New*	*Deleted*	*Changed*	*Errata*
QNX Momentics 6.2.1	*New*	*Deleted*	*Changed*	

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Neutrino 6.6?

New entries

ability **(p. 32)**

Change the ability set of the invoking process (QNX Neutrino)

calib-touch **(p. 85)**

Calibrate touchscreen

deva-ctrl-via8233.so **(p. 207)**

Sound driver for the VIA 8233 Audio controller

devc-serusb_dcd **(p. 302)**

Driver for USB-to-serial adaptors

devn-smsc9500.so **(p. 398)**

Driver for the SMSC9500 USB Ethernet dongle

devnp-asix.so **(p. 412)**

Driver for the ASIX AX88172, AX88172A, AX88178, AX88772, AX88772A, AX88772B USB Ethernet dongle

devnp-ecmplus.so **(p. 427)**

Driver for the CDC ECM/RMNET USB Ethernet control module

devnp-ixgbe.so **(p. 436)**

Driver for Intel 10 Gigabit Ethernet controllers

devnp-usbdnet.so **(p. 452)**

Class Driver for USBDNET (USB Device Network Driver)

devu-umass_client-block **(p. 470)**

Function driver for USB MASS storage devices

devu-xhci.so **(p. 474)**

Driver for Extensible Host Controller Interface (XHCI) for USB 2.0

dhclient **(p. 478)**

Copyright © 2014, QNX Software Systems Limited

Dynamic Host Configuration Protocol client

dhclient-script **(p. 487)**

DHCP client network configuration script

dhclient.conf **(p. 492)**

DHCP client configuration file

dhclient.leases **(p. 504)**

DHCP client database of acquired leases

DHCP conditional behavior **(p. 505)**

Specify conditional behavior for DHCP servers and clients

DHCP options **(p. 514)**

Dynamic Host Configuration Protocol options

dprepresize **(p. 641)**

Prepare a Power-Safe (`fs-qnx6.so`) filesystem for resizing

dresize **(p. 643)**

Resize a Power-Safe (`fs-qnx6.so`) filesystem

dvfs_client **(p. 660)**

Client for interacting with a Dynamic Voltage Frequency Scaling driver

dvfsmgr-* **(p. 663)**

Dynamic Voltage Frequency Scaling driver

fsencrypt **(p. 834)**

Filesystem encryption manager

fs-rcfs.so **(p. 827)**

Shared object that supports the read-only compressed filesystem (QNX Neutrino)

getfacl **(p. 903)**

Get the access control list for a file or files

io-usb-dcd **(p. 1018)**

Server for universal serial bus (USB)

Copyright © 2014, QNX Software Systems Limited

lsm-slip.so **(p. 1157)**

Loadable stack module that supports Serial Line IP (SLIP) network interface

mkfatfsimg **(p. 1228)**

Build a FAT filesystem image (QNX Neutrino)

mkqnx6fsimg **(p. 1279)**

Build a Power-Safe filesystem image (QNX Neutrino)

mkrcfs **(p. 1292)**

Create a read-only compressed filesystem (QNX Neutrino)

mkrcfsimg **(p. 1296)**

Create a read-only compressed filesystem image (QNX Neutrino)

pathtrust **(p. 1442)**

Designate a file or filesystem as trusted, or see if it is

plainrsa_gen **(p. 1558)**

Generator for Plain RSA keys

racoonctl **(p. 1651)**

racoon administrative control tool

slattach **(p. 1772)**

Attach serial lines as network interfaces

slm **(p. 1798)**

System launch and monitor: launch complex applications consisting of many processes that must be started in a specific order

slogger2 **(p. 1812)**

System logger

slog2info **(p. 1816)**

Display messages from the system log

startup-apic-32 **(p. 1854)**

Startup for Intel Advanced Programmable Interrupt Controller (APIC) systems, using 32-bit physical addresses

Copyright © 2014, QNX Software Systems Limited

ulink_ctrl **(p. 2004)**

Control a USB DCD link

wave **(p. 2045)**

Play back audio data

waverec **(p. 2046)**

Record audio data

Differences between DHCPv4 and v6

There are a few key non-obvious differences between the new and old DHCP that you should be aware of:

- As an extension, we've added two new command-line options to *dhclient* (p. 478):

  -m

  Write the resolver configuration to memory (using `confstr`) rather than to `/etc/resolv.conf`.

  --no-resolve

  Don't install the resolver configuration at all.

- `dhclient`: All the local node configuration is taken care of by the `dhclient-script`, i.e. users are free to change and adapt this, which gives them a lot of flexibility. We install a default version of this script in `/sbin`.
- The above -m and --no-resolve command-line options are implemented by simply passing environment variables ***RESOLV_TO_MEMORY*** and ***MODIFY_RESOLV*** respectively to the script, and the default script handles them.
- The default `dhclient-script` overwrites the local resolver configuration; it doesn't attempt to merge it. This has the advantage that stale resolver configuration isn't kept around, but also has drawbacks (see below).
- The old `dhcp.client` process would handle one interface; the new `dhclient` can handle several interfaces in one process (but the state machines are separate).
- The new client, relay, and server support IPv6 (the old ones didn't).
- The new client, relay, and server can be started in *either* IPv4 mode (default, -4 command-line option) or IPv6 mode (-6), but not both at the same time.
- If two or more instances of `dhclient` run at the same time, there are potential races for any common configuration parameters they may both receive, and this should be handled by the `dhclient-script`. One example of such configuration parameters is the resolver configuration, and the solution in that particular case is

Copyright © 2014, QNX Software Systems Limited

to run one of the clients with the --no-resolve command-line option or change the default `dhclient-script`.

- The ARP probing feature in old `dhcp.client` isn't implemented in the new `dhclient`, the place to invoke such functionality would be in the `dhclient-script`.
- BUG: Failures of IPv6 Duplicate Address Detection aren't captured by `dhclient`. This should apparently be handled by the `dhclient-script`, but currently the script's return values aren't processed, so this won't work.
- We've added support for DHCPv6 over PPP interfaces (and we hope to get ISC to officially support this patch).
- `dhclient` will not clean up if it's simply killed. Instead, use the -r and -x options.
- IAIDs are sequence numbers that start based on the MAC address for regular interfaces, but are simple sequence numbers starting at zero for PPP interfaces.
- DHCPv6 uses DHCP Unique Identifiers (DUIDs), which just need to be unique per node, not per interface as you might think. A DUID is an opaque identifier to the peer, and the peer can use only it to compare equality (==).

  If a DUID is in the leases file, it'll be grabbed from there before trying to create it. You can't form it from PPP interfaces; instead our patch tries other interfaces (if they're specified on the command line). You can also set the DUID manually, as described in the entry for *`dhclient.conf`* (p. 492).

- DHCPv4 isn't expected to work on PPP; if you try it, you'll get an error message like this:

  ```
  Unsupported device type 23 for "ppp0"
  ```

The DHCP utilities have the dependencies listed below. These dependencies are really important only if you're creating an image file; if you're booting from a full installation, they should already be present.

dhcpd

Has a dependency on the following libraries/binaries:

- `libcrypto.so`
- `libsocket.so`
- `libdhcpctl.so` (built as part of io-pkt/services/dhcp)
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)

Configuration files:

- `/etc/dhcpd6.conf` (required, DHCPv6 config file; you can override it with -cf *config_file* on startup)
- `/etc/dhcpd.conf` for IPv4 (DHCPv4) operation.

Copyright © 2014, QNX Software Systems Limited

- `/var/db/dhcpd6.leases` (required, database and server ID, needs to be read/write; you can override it with -lf *leases_file* on startup)
- `/var/db/dhcpd.leases` for IPv4.

You should generally create an empty leases file.

`dhclient`

Has a dependency on the following libraries/binaries:

- `libcrypto.so`
- `libsocket.so`
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)
- `/sbin/dhclient-script` (required; you can override it with -sf *script-file* on startup)

Configuration files:

- `/var/db/dhclient6.leases` (optional; if not present, it will generate a new client ID on every startup; you can override it on startup)
- `/etc/dhclient6.conf` (optional; defaults used if not present; you can override it on startup)
- `/etc/dhclient.conf` for IPv4
- `dhclient6.leases`
- `dhclient.leases` for IPv4

`dhclient-script`

Has a dependency on the following libraries/binaries:

- `ifconfig` (which needs `io-pkt-*` and `libsocket.so`)
- `sh`
- `route`
- `hostname`
- `getconf`
- `setconf`
- `cat`
- `mv`
- If `/etc/dhclient-enter-hooks` and/or `/etc/dhclient-exit-hooks` exists, then `dhclient-script` will run them too.

Configuration files:

Copyright © 2014, QNX Software Systems Limited

- `/etc/resolv.conf` (optional; needed to update the DNS server; must be read/write)

`dhcrelay`

Has a dependency on the following libraries:

- `libcrypto.so`
- `libsocket.so`
- `io-pkt-v4`, `io-pkt-v4-hc`, or `io-pkt-v6-hc` (depending on whether you're using IPv4 or IPv6)

All configuration is done from the startup command line.

Deprecated content

We've deprecated the following:

- support for MIPS, PowerPC, SH4, ARM9, and ARM11 targets
- self-hosted QNX Neutrino systems
- `cvs`
- `devn-asix.so` — replaced by *`devnp-asix.so`* (p. 412)
- `devn-speedo.so` — replaced by *`devnp-speedo.so`* (p. 449)
- `devnp-ath.so`
- `devnp-axe.so` — replaced by *`devnp-asix.so`* (p. 412)
- `devnp-bcm43xx.so`
- `devnp-ral.so`
- `devnp-rum.so`
- `devnp-ural.so`
- `dhcprelay` — replaced by *`dhcrelay`* (p. 615)
- `diff3`
- `diskboot`
- `enum-usb` — replaced by `usblauncher` (see the *Device Publishers Developer's Guide*)
- `finstall`
- `mksbp`
- `patch`
- `qbinaudit`
- `qde` — replaced by *`run-qde`* (p. 1723)
- `qed`
- `QWinCfg` — *`run-qde`* (p. 1723) sets up the development environment before starting the IDE

Copyright © 2014, QNX Software Systems Limited

- `setupbsp`
- SNMP:
  - `/etc/acl.conf`
  - `/etc/context.conf`
  - `/etc/mib.txt`
  - `mstrip`
  - `/etc/party.conf`
  - `smic`
  - `snmpbulkwalk`
  - `/etc/snmpd.conf`
  - `snmpd`
  - `snmpgetnext`
  - `snmpget`
  - `snmpnetstat`
  - `snmpset`
  - `snmpstatus`
  - `snmptest`
  - `snmptranslate`
  - `snmptrapd`
  - `snmptrap`
  - `snmpwalk`
  - `/etc/view.conf`

  Use a third-party solution instead.
- `svn`
- `sysinfo`
- `who`

Changed content

aps **(p. 41)**

We've deprecated the `bmp_safety` policy for the -S option.

chgrp **(p. 109)**

There's a new -h option that makes this utility modify the symbolic link instead of the referenced file.

chkqnx6fs **(p. 121)**

We've documented the -f and -S options.

Copyright © 2014, QNX Software Systems Limited

chown **(p. 129)**

There's a new -h option that makes this utility modify the symbolic link instead of the referenced file.

coreinfo **(p. 140)**

The following options are new:

- -i — Display process information.
- -l — Display the `QNT_LINK_MAP` note if present.
- -m — Display the memory map.
- -s — Display system information.
- -t — Display information about thread(s).
- -v[v...] — Be verbose.

cp **(p. 141)**

We've implemented these POSIX options:

- -H — follow symbolic links in source operands. Symbolic links found in tree traversal aren't followed.
- -L — follow symbolic links.
- -P — don't follow symbolic links.

Because of these additions, the former -L (a QNX Neutrino extension) is now -q.

devb-ahci **(p. 225)**

We've updated the options.

devc-ser* **(p. 169)**

All the `devc-ser*` drivers support a -v option that makes the driver verbose.

devc-serusb **(p. 298)**

This driver supports CDC ACM; it automatically detects ACM devices, based on the USB Class code.

devf-generic **(p. 311)**

The `devf-*` drivers now support 32- and 64-byte error-correcting code (ECC) partitions. The -x option now takes an argument that you use to specify which alignment to use.

devi-hid **(p. 339)**

Copyright © 2014, QNX Software Systems Limited

We've updated the options.

devnp-ncm.so **(p. 441)**

The ext_name option makes the driver add the bus number and device number to the network interface name.

devu-ehci.so **(p. 459)**

- There's a new en_sched option that always enables the scheduler.
- We've documented the frame_list_size, num_itd, and soft_retries options.

dhcpd **(p. 552)**

dhcpd.conf **(p. 566)**

dhcpd.leases **(p. 610)**

Updated to include support for DHCPv6.

When you invoke `dhcpd`, `dhclient`, or `dhcrelay`, the command-line options indicate whether you want to use IPv4 or IPv6. The IPv6 versions of the leases, pid, and configuration file have a "6" appended to their name:

IPv4 name	IPv6 name
`dhcpd.conf`	`dhcpd6.conf`
`dhclient.conf`	`dhclient6.conf`
`dhcpd.leases`	`dhcpd6.leases`
`dhclient.leases`	`dhclient6.leases`
`dhclient.pid`	`dhclient6.pid`
`dhcpd.pid`	`dhcpd6.pid`
`dhcrelay.pid`	`dhcrelay6.pid`

For example:

- If you start `dhcpd` with the -4 option (or without specifying -4 or -6), it uses `dhcpd.conf`, `dhcpd.leases`, and `dhcpd.pid`;
- If you start `dhcpd` with the -6 option, it uses `dhcpd6.conf`, `dhcpd6.leases`, and `dhcpd6.pid`.

Copyright © 2014, QNX Software Systems Limited

This naming scheme lets you run separate copies of `dhcpd`, `dhclient`, `dhcrelay` and so on for the IPv4 and IPv6 protocols. The only place where there's contention is in the updating of `/etc/resolv.conf`, which is done by `dhclient-script`.

---

dumper (p. 652)

The following options are new:

- -b — attempt to slog a backtrace.
- -D *path* — the same as -d, but without querying `authman`.
- -F — run at a fixed priority.
- -f — follow soft links for the creation of the dump files.
- -N *max_files* — save sequential dumps, to a maximum of the given number of files.
- -S — disable the dumping of shared memory mappings.
- -U *user_name* or -U *uid*[:*gid*[,*sup_gid*]*]] — once running, run as the specified user, so that the program doesn't need to run as `root`.

Note that `SIGDEADLK` and `SIGEMT` refer to the same signal.

We've added a sample program that registers for dump notifications.

flashctl (p. 771)

If you're using ECC partitions, you need to use the -b option to specify the appropriate alignment.

fs-dos.so (p. 795)

We've added `koi8r` to the list of supported *mapping* values for the codepage option.

fs-etfs-ram (p. 801)

- We've documented the -k and -L options.
- Note that filenames that are more than 32 bytes long use two directory entries, reducing the number of files that you can actually have.

fs-qnx6.so (p. 823)

The following options are new:

- trim — enable or disable support for TRIM, or choose to use discard instead.
- crypto — enable or disable encryption support.

Copyright © 2014, QNX Software Systems Limited

hogs (p. 939)

The following options are new or have changed:

- -i *iter* — limit the output to the specified number of iterations.
- -m [e][t][p][s] — specify the type of memory mappings you want to include in the memory total for each process.
- -S [clmlp] — sort the output by CPU (the default), memory, or process ID.
- -% *num* [clm] — show only processes that consume this percentage or more CPU (`c`, the default) or memory (`m`).

inflator (p. 984)

There's a new -t option that you can use to specify the maximum number of worker threads. The default is 4.

io-blk.so (p. 993)

In order to reduce the size of `io-blk.so`, we've moved the character sets for Japanese, traditional Chinese, and standard Chinese (code pages 932, 936, and 950) into a separate library called `charset.so`, which `io-blk.so` loads only if needs to mount a FAT volume, CD, or DVD that's formatted in one of these locales.

*io-pkt-** (p. 1007)

- The documentation now mentions that `io-pkt` supports TUN and TAP.
- We've documented the generic driver option, unit=*number*, which you can use to override the interface number.
- Note that in order to use SSL connections, you must have started *random* (p. 1654) with the -t option.
- The term “threads” in the TCP/IP threads_* options is a misnomer; it really refers to functional TCP/IP connections or blocking operations (*read()*, *write()*, *accept()*, etc.). It has nothing to do with the number of threads running in `io-pkt-*`.

io-usb (p. 1015)

- There's a new -C option that makes `io-usb` never select a configuration at enumeration time (hubs excepted).
- We've documented the -h and -m options, which specify the high watermark and the maximum number of threads in the thread pool.

Copyright © 2014, QNX Software Systems Limited

kill **(p. 1026)**

There's a new -a option that makes this utility signal all processes whose application ID matches the value of the *pid* argument.

ln **(p. 1114)**

Note that you can get some surprising results if you create a symbolic link to a directory that's a symbolic link to somewhere else. For more information, see “Symbolic links” in the Working with Filesystems chapter of the QNX Neutrino *User's Guide*.

lsm-qnet.so **(p. 1149)**

We've added more details about use Qnet over a VLAN interface.

mkefs **(p. 1209)**

- The `devf-*` drivers now support 32- and 64-byte error-correcting code (ECC) partitions. The new `ecc_on` attribute for `mkefs` lets you disable or enable ECC support. The value of the attributes indicates which alignment to use.
- There's a new `mtime` attribute that you can use to set the timestamps of files and directories.

mketfs **(p. 1219)**

There's a new `mtime` attribute that you can use to set the timestamps of files and directories.

mkifs **(p. 1241)**

- We've described the `sched_aps=`*partition_name* modifier that you can use in a startup script to launch processes in an adaptive partition.
- There's a new `mtime` attribute that you can use to set the timestamps of files and directories.

mkqnx6fs **(p. 1274)**

There's a new -E option that you can use to enable support for file data encryption.

mq **(p. 1317)**

New options:

Copyright © 2014, QNX Software Systems Limited

- -U — once established, run as the given user, specified either as a name or an ID.

omshell **(p. 1412)**

Updated to include support for DHCPv6.

on **(p. 1417)**

- You can now optionally specify a polling interval, in milliseconds, for the -W option. The default is 100 milliseconds.
- New options:
  - -A — specify an ability to be allowed or disallowed.
  - -ad | -ae — disable or enable Address Space Layout Randomization (ASLR).
  - -E — clear all environment variables, including any preceding -e options.
  - -e — define an environment variable for the child process.
  - -L — specify a limit on a system resource for the child process.

openssl **(p. 1425)**

Note that in order for `openssl` to be fully functional, you must have started *random* (p. 1654) with the -t option.

passwd **(p. 1434)**

This utility now has options that let you choose the encryption method:

- -d — DES
- -m — MD5
- -S — SHA-512 (the default)
- -s — SHA-256

The following options are also new:

- -t *iterations* — the number of times to iterate the hash.
- -w *width* — the width of the salt, in multiples of 8 bytes.

pidin **(p. 1521)**

- Note that this utility goes through all of the processes in the system, one by one, and retrieves state information for each; it doesn't get the system state in a single snapshot. Thus `pidin`'s output reflects what was

Copyright © 2014, QNX Software Systems Limited

happening at the instant in time when it queried each process, so you might see some anomalies, suchs as more threads running than there are processors in the system.

- If you aren't logged in as `root`, `pidin` can't get full information about processes owned by other users.
- There's a new backslash (\) format character that makes `pidin` display the supplementary group IDs of the user who launched the process. It's now included in the `users` shorthand form.
- There's a new `O` format character and `libs` shorthand form that make `pidin` display the loaded shared libraries.
- There's a new `]` formatting code that makes `pidin` display the process's application ID.
- The `R` formatting code and the `timers` shorthand form now display information about precise and tolerant timers.
- There's a new ^ formatting code and `tolerance` shorthand form that make `pidin` display the process's default timer tolerance.
- Note that the -n option isn't compatible with the `o` format or the extsched and fds shorthands; use `on -f` *`remote_node`* `pidin ...` instead.

`pipe` (p. 1555)

New options:

- -U — once established, run as the given user, specified either as a name or an ID.

`procnto*` (p. 1586)

- In QNX Neutrino 6.4.0, the default value for EALREADY was the same as EBUSY (corresponding to the -eo option); in QNX Neutrino 6.6.0, it's changed to a different value (corresponding to -en).
- There's a new memory-manager option, -mc, that you can use to tell the kernel to zero memory when it's freed instead of waiting until the next time the memory is allocated. You can use the -m~c option to explicitly request the default behavior.
- The -ae support for `procnto` for ARMv7 processors has been changed to allow misaligned accesses to be performed in hardware. This change means that `ThreadCtl(_NTO_TCTL_ALIGN_FAULT, ...)` no longer changes the alignment-fault behavior on a per-thread basis on these targets.
- The default `umask` used when creating the entries in `/proc/`*`pid`*`/as` is now 0066 instead of 0022.

Copyright © 2014, QNX Software Systems Limited

Opening `/proc/`*pid*`/as` for read-only access succeeds, even if the file permissions would normally say that it should fail with an EACESS. Instead the kernel marks the OCB as allowing only *devctl()* commands. This prevents unprivileged processes from examining a process's memory, but still allows a non-`root` `pidin` to display some useful information.

- There's a new memory-manager option, -mX, that makes the kernel fail any attempts by *mmap()* or *mprotect()* to turn on PROT_EXEC for a memory-mapped file mapping if the file doesn't have execute permission for the client process; an error of EACCES will be given instead. The default -m~X option allows such attempts.
- You can append an s or S to the -P *priority* option if you want out-of-range priority requests to saturate at the maximum allowed value instead of resulting in an error.

qcc*, *QCC **(p. 1608)**

There's a new -std=*language* option that gets passed to the underlying tools.

qconn **(p. 1621)**

Your target system now needs to include `libtracelog.so`.

random **(p. 1654)**

New options:

- -U — once established, run as the given user, specified either as a name or an ID.

rpcbind **(p. 1695)**

This utility needs a writable `/var/run` directory in order to run.

slogger **(p. 1807)**

New options:

- -U — once established, run as the given user, specified either as a name or an ID.

startup-apic **(p. 1854)**

There's a new -z option that makes `startup-apic` use the 8254 as system clock instead of the default High Precision Event Timer (HPET).

Copyright © 2014, QNX Software Systems Limited

startup-* options **(p. 1848)**

There's a new -C option that makes the startup library clear (i.e., zero) any memory that it allocates.

tcpdump **(p. 1895)**

Note that you must be `root` in order to read packets from a network interface. Reading a saved packet file doesn't require special privileges; you just need read permission for the file.

tracelogger **(p. 1974)**

There's a new -p option that you can use to specify the address of the physical memory to use for kernel buffers.

waitfor **(p. 2044)**

You can now optionally specify a polling interval, in milliseconds. The default is 100 milliseconds.

Errata

devb-adpu320 **(p. 217),** ***devb-aha8*** **(p. 221),** ***devb-ahci*** **(p. 225),** ***devb-btmm*** **(p. 230),** ***devb-eide*** **(p. 235),** ***devb-fdc*** **(p. 244),** ***devb-mvsata*** **(p. 253),** ***devb-ram*** **(p. 258)**

The limit on I/O (i.e., the *lseek()*, *read()* and *write()* functions) depends on the type of filesystem mounted and on whether you use the 32- or 64-bit versions of these functions. The maximum number of blocks is 2^{32}.

devb-umass **(p. 262)**

This driver also supports disk options that control the driver's interface to *cam-disk.so* (p. 91).

devnp-ecm.so **(p. 424),** ***devnp-ecm.so*** **(p. 424)**

If you specify the pnp option, then it's when you use `ifconfig` *interfaceX* `destroy` to remove the last interface that the DLL is unloaded.

fs-qnx6.so **(p. 823)**

Use commas to separate the filesystem options.

gcc **(p. 889)**

The -fwritable-strings is obsolete and was removed in GCC 4.0.

Copyright © 2014, QNX Software Systems Limited

io-audio **(p. 989)**

It's all right to start more than one instance of `io-audio`.

mkefs **(p. 1209),** ***mketfs*** **(p. 1219),** ***mkifs*** **(p. 1241)**

- The default value of the `optional` attribute is true.
- If you give a single path for a file specification, the file is copied from the host to the location in the image defined by the `prefix` attribute. If the path isn't absolute, the utility looks for it in the locations identified by the `search` attribute.
- We've corrected the description of the `followlink` and `type=dir` attributes.

ntpd **(p. 1373)**

The default location of the NTP configuration file is `/etc/ntp.conf`; there is no default symmetric key file.

ntpdate **(p. 1379)**

- The possible values of the -o option are 1, 2, 3, and 4. The default is 4.
- The default location of the symmetric key file is `/etc/ntp.keys`.

ntpdc **(p. 1382)**

The version number for the `addpeer` can be 1, 2, 3, or 4, and defaults to 3.

on **(p. 1417)**

You can't specify the sporadic scheduling policy with the -p option.

QCC, qcc **(p. 1608)**

Note that the application binary interfaces (ABIs) for the Dinkum C++ and GNU C++ libraries are incompatible. If you link a program against both libraries, you'll get a segmentation fault.

random **(p. 1654)**

The `/dev/random` and `/dev/urandom` device entries provide the same functionality.

sysctl **(p. 1875)**

The `kern.mbuf.msize` variable is read-only.

Copyright © 2014, QNX Software Systems Limited

What's new in the QNX Software Development Platform 6.5.0 Service Pack 1?

New entries

asa **(p. 48)**

Translate line-printer control sequences to newlines/form feeds (POSIX)

bzip2recover **(p. 82)**

Recover data from damaged `bzip2` files

cfgopen **(p. 98)**

Search for configuration files (QNX)

comm **(p. 136)**

Select or reject lines common to two files (POSIX)

confstr **(p. 138)**

Get or set a configuration string

csplit **(p. 161)**

Split a file based on the context (POSIX)

deva-ctrl-usb.so **(p. 203)**

Sound driver for USB audio devices

devnp-ecm.so **(p. 424)**

Driver for the CDC ECM USB Ethernet control module

devnp-ncm.so **(p. 441)**

Driver for the USB CDC NCM network control module

svn

Subversion command-line client

termdef **(p. 1949)**

Display or set the terminal type (QNX)

wiconfig **(p. 2052)**

Configure WaveLAN/IEEE devices

Copyright © 2014, QNX Software Systems Limited

wlanctl **(p. 2055)**

Examine the IEEE 802.11 wireless LAN client/peer table

Changed content

bzip2 **(p. 80)**

We've updated the list of options to reflect the current version of the software.

devb-eide **(p. 235)**

- We've added the following cam option:
  - resmgr=*m*:*l*:*h*:*d*

  as well as the following interface-specific options:
  - altstatus
  - enable
  - iwaitnbsy=*ms*
  - resets=*num*
  - tmem=*name*
  - vaddr
  - verbose=*level*

  and the following device-specific options:
  - apm_level=*level*
  - drdy=*mode*
  - nobmstr
  - rahead=*state*
  - spinup=*time*
  - verbose=*level*

  The following options are obsolete:
  - noconcurrent (interface-specific)
  - lba48 (device-specific)

  The wcache device-specific option takes a value of `on` or `off`.
- We've described how the driver chooses the connection mode.

devb-loopback **(p. 247)**

This driver has the following new options:

Copyright © 2014, QNX Software Systems Limited

heads=*num*

Specify the number of heads (default 1).

tracks=*num*

Specify the number of sectors per track (default 1).

devc-*

We've documented the following suboptions for the -o option:

- nodaemon — don't daemonize
- priority=*prio* — set the working priority of the internal pulse.

devc-serusb **(p. 298)**

We've documented the following options:

- debug=[rx|tx|intr|all|none] — write raw URB data to ***slogger*** (p. 1807)
- drt=*num* — data ready timeout
- name=*name* — the basename for the pathname entry
- retry=*num* — the number of retries if a status of USBD_STATUS_DEV_NOANSWER is given
- -o nodaemon — don't daemonize

devf-generic **(p. 311)**

We've documented the following options:

- -A — when registering the path names for the partitions with *resmgr_attach()*, use the _RESMGR_FLAG_AFTER flag to force the path to be resolved after others with the same pathname at the same mountpoint.
- -L *limit* — the number of retries to make if the physical flash erase functions for a unit fails. The default is 0.

The following options are new:

- -D — enable automatic detection of ECC mode.
- -x — enable software ECC mode.

Don't mix ECC-enabled partitions and ECC-disabled partitions; the driver doesn't support this.

Copyright © 2014, QNX Software Systems Limited

The arguments to the -t option have changed. You can now specify the high water, low water, and maximum number of threads for the driver's thread pool.

devh-usb.so **(p. 337)**

This driver has a new priority option that lets you specify the priority of the removal thread. The default priority is 8.

devi-hid **(p. 339)**

This driver has a new -R option that lets you specify the display resolution (e.g., -R800,480). If you specify this option, `devi-hid` doesn't use the graphics framework to determine the resolution.

`devn-asix.so`

This driver now supports AX88172A and AX88772B USB Ethernet dongles.

devnp-e1000.so **(p. 420)**

The following options are new:

int_mod=*N*

The interrupt moderation value. The default is 20000 interrupts/sec; a value of zero disables interrupt moderation.

force_link

Force the link speed/duplex. The default is to autonegotiate the advertised speed/duplex.

max_read=*N*

The maximum PCIe read request size. *N* must be 128, 256, 512, 1024, 2048, or 4096 bytes.

tx_reap=*N*

The maximum number of transmit descriptors to reap. The default is 64.

Other changes include:

- The default number of receive descriptors is now 512, and the maximum is 4096.
- The default number of transmit descriptors is now 4096, and so is the maximum.

Copyright © 2014, QNX Software Systems Limited

devn-pegasus.so **(p. 378)**

This driver now supports busnum and devnum options.

devnp-shim.so **(p. 447)**

This driver now has a shimrxcopy option that controls how the shim layer copies packets.

devu-ehci.so **(p. 459)**

New options:

ports=*port*:*port*...

Set the enumeration order of each root port. Use colons to separate the port numbers.

memory=*name*

Use the specified typed memory for DMA descriptors (endpoint descriptor, transfer descriptors, and so on).

devu-ohci.so **(p. 464)**

New options:

isoptd=*num*

Restrict the number of isoch frames each TD can transfer (default 8).

memory=*name*

Use the specified typed memory for DMA descriptors (endpoint descriptor, transfer descriptors, and so on).

dhcp.client **(p. 544)**

- The following options are new:

  -H

  Don't apply the hostname to the local system, whether supplied by the server or via the -h option. In the latter case, the hostname is still sent to the server via option 12.

  -k

Copyright © 2014, QNX Software Systems Limited

Don't set the `CS_DOMAIN` (Domain Name) configuration string if you're using the -m option.

- The `dhcp.client` utility now runs an optional script, `/etc/dhcp/dhcp-check`, that takes the environment as `dhcp-up` does, but returns a value indicating if the configuration is acceptable (0 means OK; other values result in a DHCPDECLINE).

dumper **(p. 652)**

We've documented these options:

- -P — dump the physical memory mappings.
- -t — dump the stack of the errant thread only, instead of for all threads.

dumpifs **(p. 658)**

We've described the -d option, which you can use to specify the directory to which to extract files.

enum-usb

The configuration file for `enum-usb` now supports a NoMSString option. Some devices don't support Microsoft-defined USB descriptors and will go haywire if queried for them. This option prevents queries from being made for these descriptors.

fdisk **(p. 734)**

This utility now supports logical partitions and multiple partitions of the same type.

fs-qnx4.so **(p. 820)**

The maximum numeric group or user ID on a QNX 4 filesystem is 65534.

fs-qnx6.so **(p. 823)**

There's a new alignio option that makes the filesystem attempt to align all reads and writes in sizes and offsets of the filesystem block size.

io-audio **(p. 989)**

There's a new global data_thread_prio option that you can use to specify the priority of the software mixer thread. The default is 25.

We've documented the following card options that apply to all sound drivers:

unit=*number*

Copyright © 2014, QNX Software Systems Limited

The card number to mount the driver as.

dindex=*number*

The device number that additional following options apply to.

play_name=*name*

The symbolic name to assign to the PCM playback device.

cap_name=*name*

The symbolic name to assign to the PCM capture device.

as well as the following memory (-m) options:

pool_size=*kbytes*

The size of the DMA memory pool to create, in KB.

pool_name=*string*

The name of a shared memory object to map and use as DMA memory pool. This object must be physically contiguous memory.

and the following global (-o) options:

intr_thread_prio=*priority*

Set the priority of the interrupt service threads. The default is 50.

sw_mixer_rate=[FAHQ|FA|L*value*]

Set the method of selecting the sampling frequency used by the PCM software mixing device if the underlying hardware device supports multiple rates.

sw_mixer_samples=*num*

Adjust the fragment size used by the software mixer to something other than the default of 2048 samples.

io-blk.so **(p. 993)**

If you specify a filesystem option (e.g., `noatime`) on a block filesystem, and then you remount the filesystem (`mount -u`), the flag is ignored. Similarly, if you mounted the filesystem as read-only and then remount it, the filesystem returns to its default setting. To maintain the settings, specify the options again using the -o option for the mount command.

Copyright © 2014, QNX Software Systems Limited

io-pkt-v4, *io-pkt-v4-hc*, *io-pkt-v6-hc* (p. 1007)

- There's a new -P option that you can use to specify the priority to use for `io-pkt`'s main thread. The default is 21.
- The TCP/IP module supports the following new options:

  pfil_ipsec

  (`io-pkt-v4-hc` and `io-pkt-v6-hc` only) Run packet filters on packets before encryption.

  timer_pulse_prio=*priority*

  The priority to use for the timer pulse.

io-usb (p. 1015)

New options:

-e *priority*

Set the priority of the enumeration thread.

-r *num*

Set the number of enumeration retries (default 3).

-t memory=*name*

Set the typed-memory name (default none, using sysram).

lsm-autoip.so (p. 1145)

- The documentation now shows how to load this module when you start `io-pkt-*`.
- By default, AutoIP no longer uses routes to direct the link-local network to use the routable IP address of the interface, nor does it route by default to the link-local IP address. If the old routing behavior is necessary, use the old option. Using routing this way was problematic depending on the peer configuration, as well as when multiple interfaces were present. The force option now has no effect unless you've also specified old.

mketfs (p. 1219)

The documentation now explains that cluster headers are added to the image file and are replaced with BSP-specific structures when you write the image to flash.

Copyright © 2014, QNX Software Systems Limited

mkqnx6fs **(p. 1274)**

There's a new -v option that lets you specify a volume name of up to 16 characters.

You can't specify both the -u and -v options.

mount **(p. 1312)**

If you specify a filesystem option (e.g., `noatime`) on a block filesystem, and then you remount the filesystem (`mount -u`), the flag is ignored. Similarly, if you mounted the filesystem as read-only and then remount it, the filesystem returns to its default setting. To maintain the settings, specify the options again using the -o option for the mount command.

mqueue **(p. 1319)**

The following option is new:

- -p *priority* — run at the given static priority.

We've also documented the following:

- -d — don't daemonize

mq **(p. 1317)**

We've documented the following:

- -d — don't daemonize

on **(p. 1417)**

There's a new section that contrasts the -f and -n options.

pf.conf **(p. 1476)**

IPv6 fragments are now supported. The following are no longer supported:

- `scrub fragment reassemble`
- `scrub fragment crop`
- `scrub fragment drop-ovl`

They're replaced by the reassemble feature, which you specify with the `set reassemble` command.

pidin **(p. 1521)**

Copyright © 2014, QNX Software Systems Limited

- The description of the `Blocked` column now mentions that if the blocking process is running on a remote node, the process ID is followed by @ and the node name.
- We've documented these format letters:

  ```
  C D F G L r t U u V v W w X x Y y Z z
  ```

  as well as the @ memory format letter, and the debug, thread, and process flags.
- If a shared object that contains text relocations is remapped as private, `pidin mem` now displays an exclamation mark (`!`) beside the name.

pipe **(p. 1555)**

We've documented the following:

- -d — don't daemonize

QCC, qcc **(p. 1608)**

- We've described the -save-temps option, which saves intermediate files created during compilation.
- We've added `c++-header` to the list of supported values for the -x option, and we've listed the extensions that are parsed to create precompiled headers.

showmem **(p. 1762)**

We've added more details about using the options and interpreting the output of the command.

slogger **(p. 1807)**

New options:

- -m — use `CLOCK_MONOTONIC` instead of the default `CLOCK_REALTIME` as the clock source.
- -u *event_id* — generate a user-string trace event for all messages received.

startup-* options **(p. 1848)**

In order to make startup faster and reduce jitter, *nanospin_calibrate()* now tries to read the calibration data from values stored in the system page. The startup for some boards includes an -o option that you can use to specify

Copyright © 2014, QNX Software Systems Limited

the calibration data (100 loop time and overhead) to store in the system page.

sysctl **(p. 1875)**

We've documented the -d, -e, -f, -M, -q, -r, and -x options.

Errata

brconfig **(p. 74)**

If you want to set up a bridge whenever you boot your system, you need to add the appropriate commands to your `/etc/rc.d/rc.local` file.

bzcat **(p. 79)**

With the upgrade to version 1.0.5 of `bzip` in QNX SDP 6.5.0, this command's name changed from `bz2cat` to `bzcat`.

cam-cdrom.so **(p. 89)**

We've corrected the options.

dhcp.client **(p. 544),** ***dhcpd*** **(p. 552)**

We've corrected the list of options for these utilities.

`enum-usb`

We've corrected the name of the usbmgr_path option.

/etc/inetd.conf **(p. 979)**

The descriptions of daemons and services in the default version of this file are commented out; uncomment the ones that you want to use in your system.

io-pkt-v4**,** ***io-pkt-v4-hc*****,** ***io-pkt-v6-hc*** **(p. 1007)**

If you want to load *lsm-pf-v4.so* or *lsm-pf-v6.so* (p. 1148), you need to specify the version that corresponds to the version of `io-pkt-*` that you're using.

ldrel **(p. 1098)**

This utility runs on all supported host OSs and targets.

link **(p. 1113)**

This utility creates a hard link; we've removed a reference to symbolic links.

Copyright © 2014, QNX Software Systems Limited

lsm-pf-v4.so*, *lsm-pf-v6.so **(p. 1148)**

You need to load the version of this module that corresponds to the version of `io-pkt-*` that you're using.

mkifs **(p. 1241)**

The form of the *compress* (p. 1247) attribute that lets you specify the compression algorithm doesn't have a leading + or - sign.

pfctl **(p. 1513)**

In order to permanently enable forwarding, set `net.inet.ip.forwarding` and/or `net.inet6.ip6.forwarding` to 1 in a file such as `/etc/sysctl.conf`, and then start `sysctl` using the -f option in your system's `/etc/rc.d/rc.local` file.

pidin **(p. 1521)**

The `R` format and the `timers` shorthand form report the time left before a timer's expiry and the timer interval in microseconds, not milliseconds.

QCC*, *qcc **(p. 1608)**

The GNU C++ library is supported on all architectures.

qcp **(p. 1623)**

This utility runs on all supported targets.

Copyright © 2014, QNX Software Systems Limited

What's new in the QNX Software Development Platform 6.5.0?

New entries

bison **(p. 64)**

General-purpose parser generator (GNU)

devb-loopback **(p. 247)**

Pseudo block device

devb-mvSata **(p. 253)**

Driver for Marvell 88SX50XX SATA interface (QNX Neutrino)

devh-microtouch.so **(p. 329)**

Driver for Microtouch EXII USB touch devices

devnp-bce.so **(p. 416)**

Driver for Broadcom BCM440x 10/100 Ethernet controllers

devnp-i80579.so **(p. 430)**

Driver for Intel Tolapai 80579 Gigabit Ethernet controllers

dnssec-dsfromkey **(p. 637)**

DNSSEC Delegation Signer resource record generation tool

dnssec-keyfromlabel **(p. 638)**

DNSSEC key generation tool

m4 **(p. 1162)**

Macro processor (GNU)

mkbuild

Build one or more IDE projects

mkcldr **(p. 1200)**

Convert CLDR text file to binary data for `libqdb_cldr.so`

/etc/moduli **(p. 1308)**

System moduli file for `sshd`

Copyright © 2014, QNX Software Systems Limited

pci-bios-v2 **(p. 1455)**

Provide support for the PCI BIOS and Message Signaled Interrupts (MSI).

pps **(p. 1569)**

Persistent Publish/Subscribe manager (QNX Neutrino)

scp **(p. 1730)**

Secure copy (remote file copy program)

sftp **(p. 1758)**

Secure file transfer program

sftp-server **(p. 1759)**

SFTP server subsystem

ssh **(p. 1838)**

OpenSSH SSH client: remote login program

ssh-add **(p. 1839)**

Add RSA or DSA identities to the authentication agent

ssh-agent **(p. 1840)**

Authentication agent

ssh-keygen **(p. 1842)**

Authentication key generation, management, and conversion

ssh-keyscan **(p. 1844)**

Gather SSH public keys

ssh-keysign **(p. 1845)**

SSH helper program for host-based authentication

~/.ssh/ssh_config, /etc/ssh/ssh_config **(p. 1841)**

OpenSSH SSH client configuration files

sshd **(p. 1846)**

OpenSSH SSH daemon

/etc/ssh/sshd_config **(p. 1847)**

Copyright © 2014, QNX Software Systems Limited

OpenSSH SSH daemon configuration file

startup-* options **(p. 1848)**

Generic options for startup programs (QNX Neutrino)

startup-apic **(p. 1854)**

Startup for Intel Advanced Programmable Interrupt Controller (APIC) systems

startup-bios-32 **(p. 1858)**

Startup for PC-compatible systems with a BIOS, using 32-bit physical addresses.

sysinfo

Gather system information for troubleshooting purposes

tsort **(p. 1994)**

Perform a topological sort of a directed graph (POSIX)

uptime **(p. 2024)**

Display the length of time that the system has been running

Deprecated content

- `deva-ctrl-cs46xx.so`
- `rftp`
- `rtelnet`

Changed content

applypatch **(p. 38)**

By default, `applypatch` now installs the host-side files only for the current host OS. There's a new -H option that makes `applypatch` install the host-side files for all host OSs.

cp **(p. 141)**

We've renamed the -n option to be -u, to be consistent with other platforms. The -n option is still recognized.

cut **(p. 166)**

This utility now supports the POSIX -b and -n options.

Copyright © 2014, QNX Software Systems Limited

devb-fdc **(p. 244)**

- Because of updates to `io-blk.so`, the minimum disk buffer cache is now 512 KB.
- The device enumerator no longer starts `devb-fdc` by default. If your system has a floppy drive, you can start the driver manually or uncomment the `devb-fdc` lines in the `/etc/system/enum/devices/block` file.

devb-ram **(p. 258)**

Because of updates to `io-blk.so`, the minimum disk buffer cache is now 512 KB.

devc-pty **(p. 288)**

The pseudo-tty manager can now support up to 256 ptys, using the naming scheme:

- `pty[p-zP-T][0-9a-f]` for the master device
- `tty[p-zP-T][0-9a-f]` for the slave device

devc-serpci **(p. 295)**

The boards that this driver supports must use PCI I/O space for the registers, and the ports must be contiguous in memory.

devf-generic **(p. 311),** ***devf-ram*** **(p. 320)**

- We've added more information about controlling timestamps with the -u option.
- The entry for `devf-generic` now includes a section that describes the information that the driver produces when you ask for *verbose output* (p. 317).

devh-egalax.so **(p. 327)**

The following options are new:

- device
- info
- noinit
- vendor

Copyright © 2014, QNX Software Systems Limited

If you're using this driver with the USB (*devh-usb.so* (p. 337)) module, you must specify the igndev option to `devh-usb.so`, specifying the Egalax vendor and device IDs.

devh-ps2ser.so **(p. 331)**

If you press several keys at once on some Microsoft keyboards, the keyboard doesn't produce any indication when you release the keys. As a result, the input driver thinks you're still holding the keys down. For more information, see *http://support.microsoft.com/kb/909528/en-us*.

devh-usb.so **(p. 337)**

We've documented the igndev option.

`devn-asix.so`

This driver now supports the AX88178 chip.

devn-dm9102.so **(p. 348),** ***devn-el509.so*** **(p. 351),** ***devn-el900.so*** **(p. 354),** ***devn-epic.so*** **(p. 357),** ***devn-fd.so*** **(p. 361),** ***devn-i82544.so*** **(p. 364),** ***devn-ne2000.so*** **(p. 370),** ***devn-tigon3.so*** **(p. 401),** ***devn-tulip.so*** **(p. 404)**

We've added the lan option.

devn-micrel8841.so **(p. 367)**

- We've added the lan and lan2 options.
- This driver supports only PCI versions of the Micrel 8841 (1 port) and 8842 (2 port) Ethernet controllers.

devn-pcnet.so **(p. 373)**

We've added the following notes to the documentation:

- This driver doesn't support Fiber PCNET cards with the AM79C971KC chip.
- We don't recommend that you use `devn-pcnet.so` on the BCM1250 platform, because it can lose receive interrupts.

devn-rtl8150.so **(p. 386)**

We've added the following notes to the documentation:

- On the SH platform, the lan= option gets overridden. As a workaround, fully specify the vendor ID, device ID, bus number, and device number to the driver when starting (e.g. vid=0x0bda,did=0x8150,busnum=1,devnum=2,lan=2).

Copyright © 2014, QNX Software Systems Limited

- This driver doesn't support multicast and promiscuous modes.

devn-tigon3.so **(p. 401)**

This network driver on the Dell PowerEdge 850 board runs only up to 100 Mbit/s, and not 1000 Mbit/s. Other boards work well at 1000 Mbit/s.

devnp-* **(p. 169)**

Native `io-pkt` and ported NetBSD drivers don't put entries into the `/dev/io-net` namespace, so a *waitfor* (p. 2044) command for such an entry won't work properly in buildfiles or scripts. Use *if_up -p* (p. 955) instead; for example, instead of `waitfor /dev/io-net/en0`, use `if_up -p en0`.

devnp-axe.so

- The version of this driver that we ship is compiled for x86 only.
- The USB-Ethernet dongle sometimes drops packets when used on slow systems or with UHCI. If you encounter problems with this driver, use the legacy `devn-asix.so` driver instead.

devnp-bcm1250.so, ***devnp-e1000.so*** **(p. 420), devnp-mpc85xx.so,** ***devnp-rtl8169.so*** **(p. 444),** ***devnp-speedo.so*** **(p. 449)**

These drivers no longer accept the promiscuous option because promiscuous mode is controlled by the stack itself. To enable promiscuous mode, use a `BIOCPROMISC` *ioctl()* command; there's currently no way to do this from the command line.

devnp-e1000.so **(p. 420),** ***devnp-i82544.so*** **(p. 432)**

The SQE (Squelch Test Errors) counter isn't applicable to these drivers, so they use it to track the number of internal Rx FIFO overruns, instead of counting them with the number of packets lost because they ran out of descriptors.

dhcp.client **(p. 544)**

- The `dhcp.client` now declines addresses if they fail an ARP probe (see RFC 2131 and RFC 5227). You can use the new -A option to set the number of consecutive ARP probe tests of the assigned address that can fail before `dhcp.client` gives up.
- We've documented the following options:

  -D ***ident***

Copyright © 2014, QNX Software Systems Limited

Specify the client identifier.

-R

Don't apply the DHCP-supplied default route.

`diskboot`

We've documented the -f, -i, and -R options.

`etfsctl` **(p. 713)**

This command controls all embedded transaction filesystems, not just those in RAM or SRAM.

New options include:

- -l *len* — the length for the subsequent -e, -R, or -r option.
- -o *offset* — the starting offset for the subsequent -e, -R, -r, -W, or -w option.
- -p — operate in software-update mode.

`fs-cifs` **(p. 790)**

We've documented the -o option, which you can use to specify various miscellaneous options concerning the handling of passwords, reconnection timeouts, and the handling of 64-bit filesystems.

`fs-dos.so` **(p. 795)**

The fatchk option is no longer supported.

`fs-etfs-*`

All embedded transaction filesystems use the same common options as *`fs-etfs-ram`* (p. 801).

`fs-qnx6.so` **(p. 823)**

We've changed the `ignore` value for the sync option to `none`, in order to make its meaning clearer. If you specify sync=none, the filesystem never issues a synchronization command to the disk, and doesn't drain dirty blocks from the filesystem cache (until an explicit *`umount`* (p. 2009)). The filesystem still supports the `ignore` value too.

`fs-udf.so` **(p. 829)**

New options include:

Copyright © 2014, QNX Software Systems Limited

- charset — specify a non-standard character set mapping for ISO 9660:1988 Primary Volumes and ISO 96660:1999 Supplementary Volumes.
- raw (replaces vcd) — set the number of raw CDDA/CDXA 2352-byte buffers and optionally the number of blocks to read with one raw I/O operation.

The blacklist for the verify option now includes some bad ISO9660:1999 SVD-mastering utilities.

fsysinfo **(p. 841)**

Some of the statistics that this utility displays have changed, because of the changes to *io-blk.so* (p. 993) (see below).

gawk **(p. 869)**

We've updated the options to reflect version 3.1.5.

gdb **(p. 892)**

If you start it from the command line on a QNX Neutrino system, gdb sets ***LD_BIND_NOW*** to 1 to force the loading of all lazy-load dependencies.

getconf **(p. 897)**

We've described the following variables:

- _SC_AIO_PRIO_DELTA_MAX
- _SC_DELAYTIMER_MAX
- _SC_GETGR_R_SIZE_MAX
- _SC_GETPW_R_SIZE_MAX
- _SC_PAGESIZE
- _SC_SEM_NSEMS_MAX
- _SC_SIGQUEUE_MAX
- _SC_THREAD_STACK_MIN
- _SC_TZNAME_MAX
- _PC_ASYNC_IO
- _PC_LINK_DIR
- _PC_PRIO_IO
- _PC_SYNC_IO

ham **(p. 928)**

To stop the HAM, you must use either the *ham_stop()* function (see the High Availability Framework *Developer's Guide*) or the *hamctrl* (p. 930) utility.

Copyright © 2014, QNX Software Systems Limited

head **(p. 935)**

In order to conform to POSIX, we've changed the -n option so that it always specifies the number of *lines* to copy. The -c option now takes an argument that specifies the number of bytes to copy. Note that the -c and -l options are QNX Neutrino extensions.

hogs **(p. 939)**

This utility gives you a *rough* idea of what's happening in your system. The documentation now describes the limitations on the data.

if_up **(p. 955)**

We've documented the -l option, which causes `if_up` to wait until the physical network link is up.

inetd **(p. 977)**

If any errors occur, `inetd` sends messages to *slogger* (p. 1807); use *sloginfo* (p. 1818) to view the system log.

io-blk.so **(p. 993)**

We've redesigned the buffer cache subsystem of `io-blk.so`, to help improve performance. The following options are no longer supported:

- bufsz — this is now hard-coded, with 512 bytes as the minimum device sector size, and 4096 as the maximum (i.e., bufsz=512:4096)
- hash — now part of the cache option
- postpone — now part of the delwri option
- noaiod
- protect — replaced by the mfu option
- wipe — replaced by the mfu option

The following options are new:

- memory
- mfu

Other changes include:

- You can specify a suffix of `g` (gigabytes) for memory arguments.
- The default delay time for delayed writes for fixed media (*delay1* for the delwri option) is now 3 seconds.
- You can now specify the sector size for the ramdisk option; the sector size was formerly 4 KB.
- You can now specify a hash size to use for the map and ncache options.

Copyright © 2014, QNX Software Systems Limited

- The default minimum for the ra option is the system page size; the default maximum is 64 times the system page size.
- The rmvto option can take a value of none, which disables removable media relearning.
- You can now specify a maximum value for the vnode option.
- The default disk buffer cache is still 15% of system RAM, but the minimum is 512 KB, and the maximum is 512 MB. If you specify an explicit size, bounds of 512 KB and 3 GB are applied.
- By default, `io-blk.so` now allocates the filesystem buffer cache (blk cache=) on affected ARM platforms from a global memory region (`SHMCTL_ANON | SHMCTL_GLOBAL`) to avoid the per-process 32 MB limitation. To override this and make the allocation from the normal `devb-*` process heap, specify blk memory=sysram.

io-pkt-v4*, *io-pkt-v4-hc*, *io-pkt-v6-hc **(p. 1007)**

- If a TCP/IP packet is smaller than the minimum Ethernet packet size, the packet may be padded with random data, rather than zeroes.
- There's a new TCP/IP pkt_typed_mem option that you can use to specify a typed memory object to allocate packet buffers from.

ksh **(p. 1029)**

The pattern `.*` now matches the `.` and `..` names.

This could cause problems with scripts that assume that the shell never matches these names.

link **(p. 1113)**

This utility now creates a hard link instead of a symbolic one.

lsm-qnet.so **(p. 1149)**

Once Qnet has a domain, you can't set Qnet to not use a domain; you can only change the domain.

mcd **(p. 1172)**

New -L option to run the MCD in local mode.

mkdir **(p. 1202)**

Note that not all filesystems support the creation of directories.

Copyright © 2014, QNX Software Systems Limited

mkefs **(p. 1209),** ***mketfs*** **(p. 1219)**

These utilities now support a `mountperms` attribute that you can use in a buildfile to specify the permissions to use for mountpoints. The default permissions are 0777 for `mkefs`, and 0755 for `mketfs`.

mkifs **(p. 1241)**

- This utility supports the following new attributes:
  - *+|-big_pages* (p. 1247) — attempt to align binaries and shared libraries at the appropriate address, based on the size of the UIP text.
  - *pagesizes=size[,size]...* (p. 1252) — define the page sizes that the underlying hardware supports, for use with the `big_pages` attribute.

  You can specify these attributes in the buildfile or in the bootfile.
- We've also documented the following:
  - *phys_align* (p. 1253) — specify the physical alignment of objects.
- The default scheduling priority and policy for processes launched from a script file are specified by *procnto* (p. 1586), and could change in future versions of QNX Neutrino. If the priority and scheduling policy of the processes are important to you, be sure to use the `pri=` modifier.
- The kernel now reaps zombies, even if you run a foreground process that doesn't exit in your OS image's startup script. However, running a foreground process that never exits still prevents the rest of the script from being executed.
- The `bios.boot` file now contains a test to determine whether or not it should behave like `bios_nokbd.boot`; it should work for the majority of platforms that previously required `bios_nokbd.boot`.

mount **(p. 1312)**

We've added some examples that show how to remount a filesystem as read-only and read-write.

nfsd **(p. 1348)**

We've documented the following options:

- -c *file* — use *file* as the exports file.
- -D — operate in debugging mode.
- -F — truncate the `subdirs` and `mntdtab` files, and then exit.
- -H *n* — specify the size of the file handle cache hash.
- -o run=foreground — run in the foreground; don't fork.

Copyright © 2014, QNX Software Systems Limited

- -p *n* — run `nfsd` on port *n*, and don't register with `portmap`.

Note that `nfsd` supports a maximum of 15 nested directory levels.

pci **(p. 1454)**

This utility is intended for debugging purposes; running it on an active system may cause some devices to hang.

pdebug **(p. 1459)**

We've documented the following options:

-1

("One") Exit `pdebug` after the debugging session is done.

-f

Run `pdebug` as a foreground process.

-n ***priority_levels***

Be nice; set the debugged program's priority to be *priority_levels* lower than `pdebug`'s.

By default, `pdebug` now sets ***LD_BIND_NOW*** to 1 to force the loading of all lazy-load dependencies. You can prevent `pdebug` from setting ***LD_BIND_NOW*** by specifying the new -l ("el") option.

pf **(p. 1461)**

We've made *ioctl()* compatible with other UNIX-based systems by enabling support for embedded pointers. This means that you no longer have to change calls to *ioctl()* into calls to *ioctl_socket()* in networking applications. As a result of this change, *ioctl()* can now indicate an error of `ENOBUFS` if there isn't enough memory available to copy the data that the embedded pointers refer to.

procnto* **(p. 1586)**

- This entry now describes `procnto-booke-smp`.
- There are new -mr and -m~r options that enable and disable address space randomization. Address space randomization is off by default; if you enable it, the kernel places certain items (e.g. the stack, `libc`) at different addresses every time you run a process. The QNX Neutrino RTOS Safe Kernel 1.0 doesn't support address space randomization.

qconn **(p. 1621)**

Copyright © 2014, QNX Software Systems Limited

We've documented the -v option, which makes qconn display its version number and then exit.

qcp **(p. 1623)**

The qcp utility works only on x86 platforms.

sed **(p. 1732)**

We've updated the options to reflect version 4.1.5.

sleep **(p. 1779)**

As a QNX Neutrino extension, the *time* argument can be a floating point number, so you can specify fractions of seconds.

startup-* (including *startup-bios* (p. 1858) and *startup-bios-32* (p. 1858))

There's a new -T option that prevents the startup program from setting the SYSPAGE_ENTRY(qtime)->boot_time field. If this field is 0 the first time you call *ClockTime()* to change the time of day, the kernel sets it to the appropriate value. This is useful if the RTC hardware isn't in UTC.

We've documented the -B option, which you can use (if the BIOS is buggy) to prevent the startup from using the ACPI table to determine the number of logical CPUs. All x86 startups support this option.

stty **(p. 1863)**

We've documented the following options:

- echoke
- echoctl
- imaxbel
- eol2
- swtch
- dsusp
- reprint
- discard
- werase
- lnext
- fwd
- rows
- par
- bits
- stopb

Copyright © 2014, QNX Software Systems Limited

syslogd **(p. 1885)**

The documentation now explains when the log entry includes a host name, and when it includes `nto` (indicating the local host).

tail **(p. 1888)**

If you use both -c and the deprecated -l option, the order of the options is important.

telnetd **(p. 1944)**

New options:

- -h — disable the printing of host-specific information before logging in has been completed.
- -U — refuse connections from addresses that can't be mapped back into a symbolic name.

tftp **(p. 1955)**

Remote filenames must start with a `/`. If a list of directories is given to `tftpd`, filenames must be in that list. To `get` a file, the file must have world-read privileges; to `put` a file, the file must have world-write privileges on the remote machine.

tracelogger **(p. 1974)**

- We've expanded the description of the modes in which you can run `tracelogger`.
- On a multicore system, if the clocks on all processors aren't synchronized, then `tracelogger` will produce data with inconsistent timestamps, and the IDE won't be able to load the trace file.

traceroute **(p. 1985)**

We've documented the -A and -a options.

traceroute6 **(p. 1991)**

We've documented the -I option.

unzip **(p. 2019),** ***zip*** **(p. 2081)**

We've updated `zip` to Info-Zip version 3.0, and `unzip` to version 6.0. These versions provide large file support.

We've documented the -P option, which you can use to specify a password to use to encrypt and unencrypt archive entries.

Copyright © 2014, QNX Software Systems Limited

The versions of these utilities that we ship are for x86 targets only. We don't ship them for Linux hosts.

Errata

applypatch **(p. 38)**

Invoke this utility as `applypatch`, not `applypatch.py`.

devb-umass **(p. 262),** ***devh-egalax.so*** **(p. 327),** ***devh-microtouch.so*** **(p. 329),** ***devh-touchintl.so*** **(p. 335),** ***devh-usb.so*** **(p. 337), devn-asix.so,** ***devu-kbd*** **(p. 462),** ***devu-mouse*** **(p. 463),** ***devu-prn*** **(p. 466)**

The default USB stack is `/dev/io-usb/io-usb`, not `/dev/usb`.

enum-usb

We've corrected the name of the cfg_file_path option in the example.

fs-dos.so **(p. 795)**

The default value of the fat option is `nonrmv`.

ifconfig **(p. 957)**

A VLAN tag is a 12-bit number, not a 16-bit number.

mkdir **(p. 1202)**

The syntax for this utility isn't the same on Windows as on QNX Neutrino.

procnto* **(p. 1586)**

We've corrected the name of the `_NTO_TCTL_IO` command for *ThreadCtl()*.

ps **(p. 1596)**

We've corrected the options and field names.

unlink **(p. 2018)**

In order to unlink a file, you need the appropriate permissions, but you don't have to be `root`.

Copyright © 2014, QNX Software Systems Limited

What's new in the QNX Software Development Platform 6.4.1?

New entries

applypatch **(p. 38)**

Install or uninstall a patch (QNX Neutrino)

devb-ahci **(p. 225)**

Driver for AHCI SATA interfaces (QNX Neutrino)

devnp-e1000.so **(p. 420)**

Driver for Intel Gigabit Ethernet controllers

devnp-rtl8169.so **(p. 444)**

Driver for Realtek 8169 Gigabit Ethernet controllers

dig **(p. 625)**

DNS domain information groper lookup utility

dnssec-keygen **(p. 639)**

DNSSEC key-generation tool

dnssec-signzone **(p. 640)**

DNSSEC zone-signing tool

ed **(p. 670)**

Text editor

`finstall`

Install a self-hosted QNX Neutrino system (QNX Neutrino)

Running this utility will destroy your current installation of QNX Neutrino. The only reason we ship `finstall` is to allow customers to create a custom installer boot image. This will likely happen only under the direction of our support staff.

fs-mac.so **(p. 809)**

Copyright © 2014, QNX Software Systems Limited

Shared object that supports Apple Macintosh HFS (Hierarchical File System) and HFS Plus (QNX Neutrino)

fs-nt.so **(p. 818)**

Shared object that supports the Windows NT filesystem (QNX Neutrino)

getty **(p. 905)**

Set terminal modes for system access (NetBSD)

host **(p. 942)**

DNS lookup utility

indent **(p. 974)**

Format C source

ldd **(p. 1097)**

List the shared libraries that a program requires (Unix)

ln-w **(p. 1118)**

Create a “link” on Windows

lwresd **(p. 1159)**

Lightweight resolver daemon

mcs **(p. 1196)**

Manipulate the comment section of an object file

named-checkconf **(p. 1333)**

Tool for checking the syntax of a named configuration file

named-checkzone **(p. 1334)**

Tool for checking a zone file

named-compilezone **(p. 1334)**

Tool for converting a zone file

nsupdate **(p. 1372)**

Dynamic DNS update utility

op **(p. 1424)**

Run a command as someone else

Copyright © 2014, QNX Software Systems Limited

patch

Use the output from `diff` et al. to update a file (GNU)

qbinaudit

Compare binaries to the officially distributed versions (QNX Neutrino)

rndc **(p. 1677)**

Name server control utility

rndc-confgen **(p. 1678)**

Key-generation tool for `rndc`

rndc.conf **(p. 1679)**

Configuration file for `rndc`

showlicense **(p. 1761)**

Display the type of QNX license that's currently active (QNX Neutrino)

top **(p. 1966)**

Display system usage (Unix)

unifdef **(p. 2014)**

Remove `ifdef`'ed C/C++ lines

Deprecated content

devn-rtl8169.so

Replaced by *devnp-rtl8169.so* (p. 444).

fs-cd.so **(p. 787)**

The *fs-udf.so* (p. 829) shared object now supports ISO-9660 filesystems in addition to UDF filesystems. You should use it instead of `fs-cd.so`.

psin

Use *pidin* (p. 1521) instead.

named-xfer

Ancillary agent for inbound zone transfers

ap

Copyright © 2014, QNX Software Systems Limited

Use *aps* (p. 41) instead.

Changed content

cam-disk.so **(p. 91)**

It's possible to specify a starting device number in the name= option.

Specify device numbers *only* on a closed system where you know all the devices and indexes.

cam-optical.so **(p. 93)**

The arguments of the timeout and verbose options have changed, and there's a new rmb option.

chkfsys **(p. 114)**

There's a new -x option that makes `chkfsys` exit with detailed error codes. Without this option, an exit status of 0 *doesn't* indicate that no problems were found with the filesystem(s).

cron **(p. 155)**

There's a new -s option that makes `cron` poll for jobs every minute (to compensate for clock skew).

devb-ram **(p. 258)**

The default amount of cache for a block I/O driver (15% of system RAM) is too high for `devb-ram`; you should use the blk cache=... option to reduce it.

devc-con, devc-con-hid **(p. 265)**

These managers support international keyboard layouts. You can use the supplied US-101 or DE-102 (German) layout, or you can define your own layout.

devf-generic **(p. 311)**

New options:

-d ***log_method***

Control the logging from the flash driver.

-e ***auto***

Copyright © 2014, QNX Software Systems Limited

Only enumerate the flash partitions, instead of doing a full scan and `mount`.

df **(p. 476)**

There's a new -h option that makes `df` display the sizes in a "human-readable" form, using bytes, KB, MB, or GB as the units.

`diskboot`

There's a new -u option that you can use to override the options passed to `io-usb`.

fdisk **(p. 734)**

- The `fdisk info` command displays a warning if the number of sectors reported by the device doesn't match the product of the number of cylinders, the sectors per track, and the number of heads. This is expected for hard drives that use zoned bit recording.
- We've added more information about partition types, the corresponding filesystems, and the commands you use to initialize the filesystems.

fs-cd.so **(p. 787),** ***fs-dos.so*** **(p. 795),** ***fs-ext2.so*** **(p. 807),** ***fs-qnx6.so*** **(p. 823),** ***fs-udf.so*** **(p. 829)**

All our disk filesystems *except* `fs-qnx4.so` use UTF-8 encoding for presentation of their filenames; attempts to specify a filename not using UTF-8 encoding will fail (with an error of `EILSEQ`) on these filesystems.

fs-qnx6.so **(p. 823)**

The Power-Safe filesystem was designed for and is intended for traditional rotating hard disk drive media. It can't guarantee power-safe robustness if the drive can't ensure that data is flushed to the storage medium. For more information, see "*Required properties of the device* (p. 825)" in the entry for `fs-qnx6.so`.

fs-udf.so **(p. 829)**

In addition to supporting UDF (OSTA-UDF/ECMA-167) filesystems, `fs-udf.so` now supports ISO-9660 (base 1998 spec, 1999 spec, Joliet extensions, Rock Ridge extensions). As a result, it has the following new options:

case=asis|lower|lupper

Copyright © 2014, QNX Software Systems Limited

Control the case used to display ISO-9660 filenames.

format=*list*

Set both the list of disk formats to support, as well as the order in which they should be probed.

info=*path*

The first character of the path can be + or -, and this controls whether empty entries (metadata descriptors not given a value) are shown in the directory or not, respectively.

perms=[*file_permissions*][:*directory_permissions*]

The permissions to use for ISO9660 files, directories, or both.

This filesystem also has a new vcd=*num* option that you can use to set the number of raw VCD 2352-byte deblocking buffers.

fsysinfo **(p. 841)**

The output now includes statistics about the number of pages mapped in and released by the heap-allocation subsystem.

ftp **(p. 848)**

We're using a newer version of `ftp`, so we've updated the link to the NetBSD documentation.

ftpd **(p. 850)**

We've restored the "Setting up a restricted `ftp` subtree" section that was removed from the documentation for QNX SDP 6.4.0.

gcc **(p. 889)**

- The -mno-fp-moves option (a QNX Neutrino extension) prevents the code generator from using floating point registers to move integers. Using floating point registers for this is very slow on systems that use floating point emulation.
- Even with exceptions disabled, the *new()* operator throws a `std::out_of_memory` exception if there isn't enough memory. If you want *new()* to return `NULL` instead of throwing an exception, overload the *new()* operator with your own.

inetd **(p. 977)**

Copyright © 2014, QNX Software Systems Limited

Use the -D option to force `inetd` to daemonize by calling *procmgr_daemon()* instead of calling *daemon()*. You need to specify the -D option to `inetd` if you're running it under the control of the High Availability Manager.

io-blk.so **(p. 993)**

- We now explain how you can use the naming= option to specify the naming scheme for devices and partitions.

  Use something other than the default (`0#`) scheme only on closed systems, and at your own risk.

- We've updated the options:
  - delwri — now specifies the delay time for delayed writes on fixed and removable media.
  - enumpart — set the order for enumerating disk partitions.
  - hash — removed.
  - nora — replaced by ra.
  - priority — set the priority of periodic filesystem callouts.
  - ra — set the minimum and maximum size of the read-ahead buffers.
  - [no]rmv — don't/do allow invalid mounts on removable media (re-insert).

io-pkt-v4*, *io-pkt-v4-hc*, *io-pkt-v6-hc **(p. 1007)**

- The stack processes a generic name= option that lets you override the default interface prefix used for network drivers.
- The stack processes the following driver options for all USB drivers using the NetBSD-to-QNX conversion library to let you identify a particular USB device using information obtained from running *usb -v* (p. 2025):
  - did=*ID* — device product ID
  - vid=*ID* — device vendor ID
  - devno=*addr* — device address, as reported by the `usb` utility
  - busno=*num* — host controller, as reported by the `usb` utility

io-usb **(p. 1015)**

You can use the -P option to specify the priority of the server.

ls **(p. 1139)**

Copyright © 2014, QNX Software Systems Limited

If you're using Qnet, doing something like `ls -R /net` can take a very long time because it recursively lists all the directories on all the machines on your network.

mkefs **(p. 1209)**

- You can specify a value of none for the filter attribute.
- You can use the -c option to specify a directory in which to cache compressed files.

mketfs **(p. 1219)**

You can specify a value of none for the filter attribute.

mkifs **(p. 1241)**

- You can specify a value of none for the filter attribute.
- If you turn compression on with the *+compress* (p. 1247) attribute, you can optionally specify the algorithm by number.

mkqnx6fs **(p. 1274)**

New options:

- -O *options* — specify boot options.
- -o *options* — specify filesystem options.

The -T option is intended to replace explicit -b, -g, -i, and -r values.

mksbp

This entry now describes how to build a System Builder project that's inside a workspace or just logically linked to one.

mount **(p. 1312)**

By default, filesystems are mounted as read-write (if the physical media permit it), but you can use the -r option to mount them as read-only.

named **(p. 1332)**

Updated to reflect BIND9.

/etc/named.conf **(p. 1335)**

Updated to reflect BIND9.

nfsd **(p. 1348)**

Copyright © 2014, QNX Software Systems Limited

If you change the exports file, you can make `nfsd` aware of the changes by either restarting it, or by sending it a `SIGHUP` signal.

ntpd **(p. 1373),** ***ntpdate*** **(p. 1379)**

For information about the associated configuration files, see *ntp.conf* and *ntp.keys* in the FreeBSD documentation.

on **(p. 1417)**

We've documented the -P option, which spawns the process, setting the SPAWN_PADDR64_SAFE flag to indicate that the process is known to operate safely with 64-bit addressing or doesn't care about the physical memory location.

pci-bios **(p. 1455)**

All PCI servers have an -x option that prevents the server from removing devices from the PCI bus while enumerating them.

pidin **(p. 1521)**

- This utility now supports the following shorthand names:
  - backtrace — display a backtrace of the calling routines for each thread in the displayed processes.
  - channels — display the lengths of the send, receive, reply and pulse queues.
  - mapinfo — show information about memory mappings.
- We've added more details about the output of the memory argument.
- The `pidin syspage` command now displays the CPU-dependent, *mdriver*, and *pminfo* sections of the system page.

pppd **(p. 1560)**

The ccp option is no longer supported because CCP (Compression Control Protocol) negotiation is enabled by default. The vj option (for enabling Van jacobson style TCP/IP header compression) is also no longer supported. QNX Neutrino supports multilink PPP.

procnto* **(p. 1586)**

- The Process Manager component of `procnto` implements a `/proc` filesystem that lets you access and control every process and thread running within the system. For more information, see “Controlling

Copyright © 2014, QNX Software Systems Limited

processes via the `/proc` filesystem" in the Processes chapter of the QNX Neutrino *Programmer's Guide*.

- New options controlling the defragmentation of physical memory:

  -ma

  Automatically mark memory pages that have a *mem_offset()* performed on it as unmovable when defragmenting physical memory.

  -m~a

  Disable the automatic marking of memory pages as unmovable (the default).

  -md

  Enable the defragmenting of physical memory (the default).

  -m~d

  Disable the defragmenting of physical memory.

  For more information, see "Defragmenting physical memory" in the Process Manager chapter of the *System Architecture* guide.

QCC **(p. 1608)**

Even with exceptions disabled, the *new()* operator throws a `std::out_of_memory` exception if there isn't enough memory. If you want *new()* to return `NULL` instead of throwing an exception, overload the *new()* operator with your own.

qconfig **(p. 1618)**

We've explained why you should use `QWinCfg` instead of `qconfig` on Windows.

`QWinCfg`

Click the **Show Packages..** button to display a list of the QNX packages that you've installed on your system.

setkey **(p. 1750)**

We've updated the list of supported algorithms. This utility now supports the `3des-deriv` encryption algorithm.

tail **(p. 1888)**

Copyright © 2014, QNX Software Systems Limited

The options for this utility now conform to POSIX; to copy a given number of bytes, use the -c *number* option.

tftp **(p. 1955)**

There's a new -e option and *port* argument, and several new commands:

- `blksize` *blk-size*
- `tout`
- `tsize`

tr **(p. 1971)**

You can specify ranges of characters with or without square brackets.

traceroute **(p. 1985)**

The documentation now includes these options:

- -d — turn on socket-level debugging.
- -P — set the "don't fragment" bit and use the next hop MTU each time a "need fragmentation" error is received, thus probing the path MTU.

wpa_supplicant **(p. 2065)**

We've updated the documentation to reflect the NetBSD version 5.0 of this daemon.

GNU binutils (***addr2line*** **(p. 34),** ***ar*** **(p. 45),** ***c++filt*** **(p. 84),** ***gprof*** **(p. 913),** ***ld*** **(p. 1096),** ***nm*** **(p. 1358),** ***objcopy*** **(p. 1406),** ***objdump*** **(p. 1407),** ***ranlib*** **(p. 1657),** ***readelf*** **(p. 1660),** ***size*** **(p. 1771),** ***strings*** **(p. 1861),** ***strip*** **(p. 1862))**

For detailed documentation about these utilities, see the GNU website at *http://www.gnu.org/*.

We once again support MIPSBE and MIPSLE targets.

Errata

arp **(p. 46)**

We've updated the documentation to reflect the currant usage message.

cam-optical.so **(p. 93)**

We've corrected the syntax of the retries and translation options.

devc-con **(p. 265),** ***devc-con-hid*** **(p. 265),** ***devc-par*** **(p. 286),** ***devc-serpci*** **(p. 295)**

Copyright © 2014, QNX Software Systems Limited

These drivers are for x86 targets only.

devc-serzscc **(p. 306)**

This driver is for x86 targets only.

devn-* **(p. 169)**

Legacy `io-net` drivers create entries under `/dev/io-net`, not under `/dev/io-pkt`.

devn-el509.so **(p. 351),** ***devn-sis9.so*** **(p. 390)**

You can use these drivers with any variant of `io-pkt`, not just `io-pkt-v4`.

devnp-i82544.so **(p. 432)**

- The default for the irq_thresh option is 9000.
- The default for the transmit option is 4096.
- This driver doesn't support the promiscuous option; the only way to enable promiscuous mode is by issuing an *ioctl()* command.

devnp-mpc85xx.so, devnp-mpcsec.so

These drivers are shipped only with the BSPs that need them.

dhcp.client **(p. 544)**

We've updated the documentation to reflect the currant usage message.

dloader **(p. 633)**

We've updated the description of ***QNX_TARGET***.

enum-devices

When you're specifying a device, `vend` should actually be `ven`.

fs-qnx6.so **(p. 823)**

We've corrected the description of the sync option.

/etc/inetd.conf **(p. 979)**

A server doesn't need to explicitly leave the master socket open when it exits.

ld **(p. 1096)**

We've corrected the description of the ***LD_PRELOAD*** environment variable.

ldrel **(p. 1098)**

Copyright © 2014, QNX Software Systems Limited

The -S option adds a note that specifies the maximum (not minimum) stack size. If you don't specify -L, the stack is specified as non-lazy.

mkifs **(p. 1241)**

We've removed a reference to `vmware.boot`, which we no longer ship.

`ntomulti-*` variants

We've removed references to the `ntomulti-*` variants (which we no longer ship) of *`addr2line`* (p. 34), *`ar`* (p. 45), *`gprof`* (p. 913), *`nm`* (p. 1358), *`objcopy`* (p. 1406), *`objdump`* (p. 1407), *`size`* (p. 1771), *`strings`* (p. 1861), and *`strip`* (p. 1862).

pf **(p. 1461)**

Although the NetBSD documentation talks about *ioctl()*, you should use *ioctl_socket()* instead in your packet-filtering code. With the microkernel message-passing architecture, *ioctl()* calls that have pointers embedded in them need to be handled specially. The *ioctl_socket()* function will default to *ioctl()* for functionality that doesn't require special handling.

pppd **(p. 1560)**

Some of the options aren't specific to QNX Neutrino, but aren't described in the NetBSD documentation.

uesh **(p. 1998)**

Starting in QNX SDP 6.4.0, `uesh` can accept a script file as an argument. The micro-embedded shell doesn't support filename expansion, so * and ? aren't special characters.

Copyright © 2014, QNX Software Systems Limited

What's new in the QNX Software Development Platform 6.4.0?

New entries

ap

Create, modify, and query adaptive partitions for the thread scheduler and memory allocator.

brconfig **(p. 74)**

Configure network bridge parameters

chattr **(p. 107)**

Manipulate the attributes of a file (QNX Neutrino)

chkqnx6fs **(p. 121)**

Check an entire Power-Safe filesystem for consistency (QNX Neutrino)

deva-ctrl-intel_hda.so **(p. 197)**

Sound driver for the Intel High Definition Audio controllers

deva-mixer-hda.so **(p. 215)**

Mixer DLL for High Definition Audio codecs

devc-serpci **(p. 295)**

Driver for serial PCIs

devc-serusb **(p. 298)**

Driver for USB-to-serial adaptors

devh-egalax.so **(p. 327)**

Driver for USB Egalax touch devices

devh-touchintl.so **(p. 335)**

Driver for USB Touch International touch devices

devn-asix.so

Driver for the ASIX AX88172/AX88772 USB Ethernet dongle

devn-micrel8841.so **(p. 367)**

Copyright © 2014, QNX Software Systems Limited

Driver for Micrel 8841 (1 port) or 8842 (2 port) Ethernet controllers

devn-rtl8169.so

Driver for Realtek 8169 Gigabit Ethernet controllers

devnp-ath.so

Driver for wireless network adapters based on the Atheros AR5210, AR5211, AR5212, and AR5213 chips

devnp-axe.so

Driver for USB (2.0) Ethernet adapters based on the ASIX AX88172 chip

devnp-bcm1250.so

Driver for Broadcom BCM1250 10/100/1000 Mbit Ethernet controllers

devnp-bcm43xx.so

Driver for the Broadcom-based 802.11b/g wireless Ethernet controller

devnp-bge.so **(p. 418)**

Driver for Broadcom 57xx Tigon3 10/100/1000 Mbit Ethernet controllers

devnp-i82544.so **(p. 432)**

Driver for Intel 82540, 82544, 82545, 82546, and 82547 Gigabit Ethernet LAN adapters

devnp-mpc85xx.so

Driver for Freescale MPC85XX TSEC Ethernet controllers

devnp-mpcsec.so

Hardware Crypto Engine driver

devnp-msk.so **(p. 439)**

Driver for Marvell Yukon-2 based Gigabit Ethernet adapters

devnp-ral.so, devnp-ural.so

Driver for wireless adapters based on the Ralink RT2500, RT2501, RT2600, and RT2500USB chipsets

devnp-rum.so

Driver for USB 2.0 wireless adapters based on the Ralink RT2501USB and RT2601USB chipsets

devnp-shim.so **(p. 447)**

Copyright © 2014, QNX Software Systems Limited

"Shim" driver for backward compatibility with `io-net`

devnp-speedo.so **(p. 449)**

Driver for Intel 82557, 82558, and 82559 Fast Ethernet LAN adapters

enum-usb

Enumerate devices on the USB bus

fs-qnx6.so **(p. 823)**

Shared object that supports the Power-Safe filesystem (QNX Neutrino)

fs-udf.so **(p. 829)**

Shared object that supports Universal Disk Format (OSTA-UDF/ECMA-167) filesystems

fsysinfo **(p. 841)**

Display filesystem statistics (QNX Neutrino)

ham **(p. 928)**

High-availability manager

hamctrl **(p. 930)**

Control a high-availability manager

hostapd **(p. 943)**

Authenticator for IEEE 802.11 networks

ifwatchd **(p. 971)**

Watch for addresses added to or deleted from interfaces and call up/down-scripts for them

lsm-autoip.so **(p. 1145)**

AutoIP negotiation module for link-local addresses

lsm-pf-v4.so, ***lsm-pf-v6.so*** **(p. 1148)**

Provide IP filter services

lsm-qnet.so **(p. 1149)**

Transparent Distributed Processing (native QNX network) module

We added the following options:

Copyright © 2014, QNX Software Systems Limited

- enforce_crc — discard packets that don't have a valid software-level CRC generated by the remote node.
- max_num_l4s — the number of interfaces.
- max_tx_bufs — the number of npkts to cache internally for transmission.
- mtu_en — the maximum transmission unit (MTU) of a Qnet packet.
- no_slog — don't send error messages to `slogger`
- qos_per_pri, qos_tx_pri — the priority of the pulses for the QoS periodic transmission thread and the QoS transmission thread.
- res_retries — the number of times the Ethernet resolver retries to resolve a node.
- res_ticks — the number of periodic ticks before the Ethernet resolver retransmits a node-resolution request.
- vtag — insert a four-byte `vlan` tag into a packet.

The combination of bind=ip and resolve=file isn't supported.

mcd **(p. 1172)**

Media Content Detector utility

mkqnx6fs **(p. 1274)**

Format a Power-Safe filesystem (QNX Neutrino)

/etc/nsswitch.conf **(p. 1369)**

Name-service switch configuration file. This file replaces the `lookup` keyword in `/etc/resolv.conf`.

openssl **(p. 1425)**

Command-line tool for using the OpenSSL crypto library

paste **(p. 1439)**

Merges lines of input files, and writes the resulting lines to standard output. (POSIX)

pf **(p. 1461)**

Packet Filter pseudo-device

pf.conf **(p. 1476)**

Configuration file for `pf`

pfctl **(p. 1513)**

Control the packet filter (PF) and network address translation (NAT) device

Copyright © 2014, QNX Software Systems Limited

pppoectl **(p. 1563)**

Display or set parameters for a PPPOE interface

python **(p. 1603)**

Object-oriented programming and scripting language

showmem **(p. 1762)**

Display memory information

showmount **(p. 1766)**

Display memory information

sockstat **(p. 1824)**

List the open sockets

tcpdump **(p. 1895)**

Dump traffic on a network

tracelogger **(p. 1974)**

We've added the -A, -c, -P, and -R options and updated the descriptions of the other options.

traceprinter **(p. 1980)**

We've added the -o option.

uudecode **(p. 2036)**

Decode a file that was encoded with uuencode

uuencode **(p. 2038)**

Encode a binary file or standard input into ASCII

who

List the logged-in users

wpa_cli **(p. 2059)**

WPA command-line client

wpa_passphrase **(p. 2064)**

Set WPA passphrase for a SSID

wpa_supplicant **(p. 2065)**

Copyright © 2014, QNX Software Systems Limited

Wi-Fi Protected Access client and IEEE 802.1X supplicant

Deprecated content

Instead of using:	Use:
`crttrap`	N/A
`flashcmp`	*`deflate`* (p. 183) and *`inflator`* (p. 984)
`fontsleuth`	N/A
`fontview`	N/A
`icc`	N/A; no longer shipped
`info`	N/A; no longer shipped
`io-net`	*`io-pkt*`* (p. 1007)
`ipf`, `ipfs`, `ipfstat`, `ipmon`, `ipnat`	*`pf`* (p. 1461), *`pf.conf`* (p. 1476), *`pfctl`* (p. 1513)
`lsm-ipfilter-v4.so`, `lsm-ipfilter-v4.so`	*`lsm-pf-v4.so`*, *`lsm-pf-v6.so`* (p. 1148)
`lsm-sctp.so`	N/A; not currently supported by `io-pkt*`
`mmplay`	QNX Aviage Multimedia Suite
`netfront`	Web Browser TDK
`nfm-autoip.so`	*`lsm-autoip.so`* (p. 1145)
`npm-pppmgr.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-pppoe.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-qnet.so`, `npm-qnet-l4_lite.so`	*`lsm-qnet.so`* (p. 1149)
`npm-qnet-compat.so`	N/A
`npm-tcpip.so`	N/A
`npm-tcpip-v4.so`, `npm-tcpip-v6.so`	Now included in *`io-pkt*`* (p. 1007)
`npm-ttcpip.so`	N/A
`phplay`	QNX Aviage Multimedia Suite
`phrecord`	QNX Aviage Multimedia Suite
`playaudio`	QNX Aviage Multimedia Suite
`playaudiocd`	QNX Aviage Multimedia Suite

Copyright © 2014, QNX Software Systems Limited

Instead of using:	Use:
`psin`	*pidin* (p. 1521), `sin`
`qnxplayer`	QNX Aviage Multimedia Suite
`voyager`	Web Browser TDK
`vserver`	Web Browser TDK

- For information about the replacement for `packagebsp`, see

  http://community.qnx.com/sf/wiki/do/viewPage/projects.bsp/wiki/Packaging_BSP

  on our Foundry27 community website.
- We've deleted the entries for the following drivers, some of which are for unsupported or obsolete boards:
  - `deva-ctrl-cyberpro5.so`
  - `deva-ctrl-tahoe.so`
  - `devb-aha2`
  - `devb-aha4`
  - `devb-aha7`
  - `devb-amd`
  - `devb-ncr8`
  - `devc-amctap`, `devc-tamctap`
  - `devc_amctap_host`
  - `devc-hspi`
  - `devc-netrom540`, `devc-tnetrom540`
  - `devc-ser2681`, `devc-tser2681`
  - `devc-ser403`, `devc-tser403`
  - `devc-ser8250-ixp2400`, `devc-tser8250-ixp2400`
  - `devc-serdsiu`, `devc-tserdsiu`
  - `devc-sergt64260`
  - `devc-serppc800`, `devc-tserppc800`
  - `devc-serpsc`
  - `devc-sersci`
  - `devc-tcon`
  - `devc-tser8250`
  - `devc-tserzscc`
  - `devg-banshee.so`
  - `devg-igs5000.so`
  - `devg-mach64.so`

Copyright © 2014, QNX Software Systems Limited

- devg-matrox.so
- devg-mq200.so
- devg-neomagic.so
- devg-orchid.so
- devg-pxa250.so
- devg-q2sd.so
- devg-ravin.so
- devg-rotate90.so
- devg-rotate270.so
- devg-rpxlite.so
- devg-s3.so
- devg-sa1110.so
- devg-stpc.so
- devg-vga.so
- devgt-iographics
- devi-ahl
- devi-carrol
- devn-artesyn.so
- devn-bcm43xx.so
- devn-cpci-mcp750.so
- devn-eepro.so
- devn-el589.so
- devn-gt64260.so
- devn-klsi.so
- devn-lance.so
- devn-ne2000-403.so
- devn-ns83815.so
- devn-orinoco.so
- devn-ppc405.so
- devn-ppc800-ads.so
- devn-ppc800-cllf.so
- devn-ppc800-mbx.so
- devn-ppc800-rpxlite.so
- devn-ppc8260.so
- devn-ppc860_mii.so
- devn-prism.so
- devn-rlan2.so
- devn-tulip-p5064.so
- devn-wd.so

Copyright © 2014, QNX Software Systems Limited

- pci-artesyn440
- pci-artesyn750fx
- pci-brh
- pci-cpc700
- pci-ixc1100
- pci-ixp2400
- pci-mpc8266
- pci-p5064
- pci-ppc440rb
- pci-raven
- pci-sandpoint
- pci-yellowknife
- startup-403evb
- startup-603evb
- startup-79s465
- startup-800fads
- startup-8260ads
- startup-artesyn
- startup-artesyn750fx
- startup-aspen
- startup-bigsur
- startup-cpci-6750
- startup-ddb-vrc5074
- startup-integrator
- startup-malta
- startup-mbx800
- startup-mcp750
- startup-mcpn765
- startup-mtx600
- startup-mvme
- startup-mvp
- startup-p5064
- startup-rpx-lite
- startup-sa1100-mm
- startup-sa1110-db
- startup-sandpoint
- startup-sengine
- startup-vme603
- startup-vr41xx

Copyright © 2014, QNX Software Systems Limited

    - `startup-walnut`

- We've removed the entries for board-specific `devf-*` drivers. All the flash filesystem drivers use the same options as *`devf-generic`* (p. 311).
- We've removed the entries related to the package filesystem:
    - `fs-pkg`
    - `packager`
    - `pkgctl`
    - `qnxinstall`
- We no longer support ARMBE, MIPSBE, or MIPSLE targets.

Changed content

`cam-cdrom.so` **(p. 89),** ***`cam-disk.so`*** **(p. 91),** ***`cam-optical.so`*** **(p. 93)**

We've documented the retries, timeout, and verbose options.

`devf-generic` **(p. 311),** ***`devf-ram`*** **(p. 320)**

The maximum number of threads that you can specify with the -t option has increased from 4 to 100.

`devn-*`

We've added some information about hardware checksumming.

`devn-i82544.so` **(p. 364)**

We've documented the pauseignore and pausesuppress options and updated the default values for the receive and transmit options.

`devn-speedo.so`

We've added the probe_phy option, which you can use to enable or disable the probing of the PHY device.

`df` **(p. 476)**

This utility rounds its figures into 512- or 1024-byte blocks (depending on the options), and it always rounds down. If the filesystem doesn't use a block size that's a multiple of 512 bytes, some rounding errors will occur.

`dinit` **(p. 626)**

We recommend that you use `dinit` to initialize a QNX 4 filesystem, and *`dloader`* (p. 633) to make it bootable. The `dinit` bootloader options are for backwards compatibility reasons, but aren't generally used anymore.

Copyright © 2014, QNX Software Systems Limited

dumper **(p. 652)**

There's a new -z option that makes `dumper` compress the core files.

`enum-devices`

- The `start`, `requires`, and `driver` clauses now support a `/wait` option that makes the enumerator pause until the command associated with the clause terminates.
- The macro for starting core networking is now `IOPKT_CMD`.

flashctl **(p. 771)**

There's now a section that describes the information that's displayed if you specify the -i option.

fs-cd.so **(p. 787)**

We've added these options:

- case — control how ISO-9660 filenames should be displayed. The case option can now have a value of `asis`.
- exe — set execute permission (on all non-RRIP regular files).
- nohsf — disable High Sierra format.

fs-dos.so **(p. 795)**

The following options have changed:

- case — new
- codepage — these names are also used for the volume label
- compat — supports a value of `os2`
- fat — new

fs-etfs-ram **(p. 801)**

An ETFS filesystem is no longer mounted by default; you can use the -m option or `mount -tetfs /dev/etfs2` *my_mountpoint*.

fs-qnx4.so **(p. 820)**

We've added the following options:

- bitmap — when to pre-read `.bitmap` files.
- grown — allow persistent over-grown files (sticky `O_APPEND` allocation).

Copyright © 2014, QNX Software Systems Limited

We've deprecated the rmvbmap option; it's equivalent to bitmap=always.

ftp **(p. 848),** ***ftpd*** **(p. 850)**

We now use the NetBSD 4.0 version of these programs, although `ftpd` also supports the -n option that was added after version 4.0.

gzip **(p. 921)**

You can now use `gzip` to compress or expand files in a RAM filesystem (`/dev/shmem`), but you need to specify the -f option.

ifconfig **(p. 957)**

Updated to work with `io-pkt*`.

io-blk.so **(p. 993)**

- The alloc option now specifies an allocation mode instead of an amount of memory to allocate.

  The default values of the following options have changed:

  - bufsz — now `512:8K`
  - fdinfo — now `always`
  - thread — now `12:2:5`

- The optional *level* argument to the verbose option indicates which categories of events to log.
- We've added the marking option.

ksh **(p. 1029)**

You can now use the **Tab** key to complete the names of files and commands.

mkifs **(p. 1241)**

- We've added the `cpu` modifier to the description of script files.
- The `/usr/lib/ldqnx.so.2` symbolic link should now point to `/proc/boot/libc.so.3`, and you should include `libc.so.2` in the list of binaries *before* `libc.so`.
- The documentation now describes the -s option and the *+|-page_align* (p. 1252) attribute.
- We've updated the default linker specification.

Copyright © 2014, QNX Software Systems Limited

mount **(p. 1312)**

We've documented the -a option.

mq **(p. 1317)**

The `/dev/mq` directory doesn't appear until you actually create a queue.

mqueue **(p. 1319)**

The `/dev/mqueue` directory doesn't appear until you actually create a queue.

nicinfo **(p. 1355)**

This utility has been updated to work with `io-pkt*`.

The documentation now describes the -c, -g, and -s options.

pidin **(p. 1521)**

We've added more details about the information that the fds shorthand gives.

ping **(p. 1543)**

This utility has been updated to work with `io-pkt*`.

If a name server isn't responding, there's a timeout of 1.5 minutes per name server.

pppd **(p. 1560)**

For information about this daemon (including exit codes), see the NetBSD documentation. We've documented the options that are specific to Neutrino.

procnto* **(p. 1586)**

- If you're using an SMP version of `procnto`, you can use the appropriate `startup-*` command's -P option to specify the maximum number of CPUs to activate.
- There's a new `procnto-v6` version of the kernel that supports ARMv6 processors.
- New options:
  - -en and -eo — control the value of EALREADY, which is changing so that it will be POSIX-compliant. For more information, see "Changes to EALREADY" in the entry for *errno* in the Neutrino *C Library Reference*.
  - -mP — turn on full allocation of high memory for all processes. This is mostly useful only for testing.

Copyright © 2014, QNX Software Systems Limited

- -m~P — make sure that all anonymous allocation occurs below the 4 GB mark (the default).
- -m[~]v — enable or disable variable page sizes. They're enabled by default.
- -m[~]x — enable or disable the PROT_EXEC flag for system-allocated threads. It's enabled by default.
- -T — specify the number of seconds to wait for a *close()* to succeed in the event of process termination.
- -u — specify the `umask` to use when creating the entries in `/proc/`*pid*`/as`.

qcc **(p. 1608)**

The -M option to `qcc` isn't changing to -Map as we warned in earlier releases; `qcc` continues to use -M for generating a mapfile.

qconfig **(p. 1618)**

This utility doesn't list the installed packages in any particular order.

qde

We no longer ship the Neutrino-hosted IDE, so `qde` now runs only on Linux and Windows development hosts.

random **(p. 1654)**

This utility creates `/dev/urandom` as well as `/dev/random`.

rtsold **(p. 1720)**

This daemon now has an -a option that lets you autoprobe the outgoing interface.

ruptime **(p. 1724),** ***rwho*** **(p. 1726),** ***rwhod*** **(p. 1727)**

The data files that these programs use are now in `/var/rwho` instead of `/usr/spool/rwho`, to conform to the Filesystem Hierarchy Standard.

rwhod **(p. 1727)**

This daemon now has -i and -u options for setting the broadcast interval and the user to run as.

setkey **(p. 1750)**

Updated to work with `io-pkt*`.

slay **(p. 1774)**

Copyright © 2014, QNX Software Systems Limited

If you change the runmask for a process, the processor for blocked threads doesn't change until the threads become unblocked (or never if the threads remain blocked).

slogger **(p. 1807)**

There's a new -c option that you can use to open the log file with O_SYNC to forcibly commit the logged events to the disk.

sysctl **(p. 1875)**

The available variables depend on what you're running on your machine; we've described the ones that you're most likely to be interested in.

tftpd **(p. 1958)**

We now use the NetBSD 3.0 version of this daemon, so the options have changed.

tinit **(p. 1964)**

- We've added the -f and -t options.
- We've described the way that `tinit` parses its configuration file.

which **(p. 2050)**

We've added the -s option, which makes the utility search for shared objects in the directories identified by the ***LD_LIBRARY_PATH*** environment variable and the _CS_LIBPATH configuration string.

Errata

chkfsys **(p. 114)**

This utility doesn't prevent itself from operating when files are open for writing on the drive.

crontab **(p. 157)**

If you want the output from your commands, redirect it to a file.

devf-* **(p. 169)**

- If you specify the -V option, the driver displays the filesystem and MTD version information, and then exits.

Copyright © 2014, QNX Software Systems Limited

- The *bwidth* and *ileave* values for the -s option must be powers of 2, but you don't specify them as powers of 2. For example, if the width of the data bus is 8, specify a *bwidth* of 8, not 3 (for 2^3).

etfsctl **(p. 713)**

Existing IPLs and bootloaders can't boot from an image in an embedded transaction filesystem.

find **(p. 750)**

We've corrected the description of the %a and %A formatting codes.

lpr **(p. 1128)**

The *printer* argument to the -P option must be a printer name that's defined in `/etc/printcap`.

mkifs **(p. 1241)**

- You can use `mkifs` to build nonbootable images. For an example, see the *Making Multiple Images* technote.
- You have to specify both `image` and `ram` file attributes if you want to create the image in ROM/FLASH; otherwise the process manager assumes that the image is in RAM.
- The `mkifs` utilities no longer strips the `QNX_usage` (usage message), and `QNX_info` (build properties) sections by default. You can use the -s option to specify additional sections not to be stripped.
- Startup code can be decompressed in place.
- Attributes that you specify with `attr=`*image_attribute* in the bootfile are processed after the -l ("el") command-line options and the buildfile, but you normally use the ? prefix on the *image_attribute*, so that it doesn't override anything explicitly set by the -l option or the buildfile.

ntpd **(p. 1373)**

After the machine has synchronized to a NTP server, the operating system time gets synchronized and corrected from time to time. This doesn't set the hardware clock; you can use the *rtc* (p. 1714) utility to set the time on the chip.

pidin **(p. 1521)**

If you use `pidin thread` or `pidin -F%h` to display the thread names, and a thread doesn't have a name, `pidin` displays the thread's ID (*tid*) instead.

Copyright © 2014, QNX Software Systems Limited

procnto* **(p. 1586)**

- The example implied that using the -p option disables preemption; it actually disables preemption only in the kernel code.
- Specifying the -as option on SH platforms is the same as specifying -ad, not -ae.

random **(p. 1654)**

If an error occurs, `random` sends a message to `slogger`, not to *stderr*.

renice **(p. 1661)**

This utility changes the priority of *all* the threads in the specified process or processes.

slogger **(p. 1807)**

We've corrected the description of `/dev/console`, what happens when multiple applications open `/dev/slog` for reading, and the example of alternating between files.

`startup-*`

The -R option can include an alignment as well as a size.

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Momentics 6.3.2?

New entries

devb-umass **(p. 262)**

- There's a new iface= option for specifying the interface number of the device.
- There's a new ignore_csw option that makes the driver ignore the Command Status Wrapper.

devc-con-hid **(p. 265)**

This console and keyboard I/O manager is similar to `devc-con`, but works in conjunction with ***io-hid*** (p. 1005) and supports PS2, USB, and all other human-interface devices.

`devc-serpsc`

PSC UART serial communications manager for MPC5200

devn-rtl8150.so **(p. 386)**

Driver for SMC2208 USB/Ethernet adapters

`packagebsp`

Package a board support package (QNX Neutrino)

`setupbsp`

Set up a Board Support Package (QNX Neutrino)

startup-bios **(p. 1858)**

The -I option is now documented. You can use this option to specify the highest-priority hardware interrupt.

use **(p. 2027)**

There's a new -s option that you can use to display the version numbers of the source used in the executable.

Changed content

`devc-sersci`

Copyright © 2014, QNX Software Systems Limited

Added the default values for the -r option for SH7770 and SH7780.

devu-prn **(p. 466)**

Added the -m option.

flashctl **(p. 771)**

This utility rounds the values of the -o and -l ("el") options down to the nearest block bound. If the range specified exceeds the partition size, it's rounded down to fit. If you use the -v option, `flashctl` displays what the values have been rounded to.

io-blk.so **(p. 993)**

- Added the before and after options.
- If you specify the verbose option, the extra information is sent to the system logger. The optional *level* argument indicates which categories of events to log.

mkifs **(p. 1241)**

This entry now describes the main bootfiles, including `raw.boot`, `elf.boot`, `binary.boot`, and `srec.boot`; see the "*Bootfile* (p. 1262)" section.

You might need to use the `perms` attribute to specify execute, setuid, and setgid permissions if you're running `mkifs` on a Windows host.

mount **(p. 1312)**

Added the before and after options, and added `io-usb` to the table of mount types.

qconfig **(p. 1618)**

Added the -c option.

pidin **(p. 1521)**

The CPU usage that `pidin times` reports are approximate, and can be inaccurate.

rpcbind **(p. 1695)**

This utility needs `/etc/netconfig`, the `librpc` shared library, as well as some specific entries in `/etc/services`.

setuid utilities

Copyright © 2014, QNX Software Systems Limited

The following utilities need to have the setuid bit set in their permissions: *crontab* (p. 157); *dhcp.client* (p. 544); fontsleuth; ; ; *login* (p. 1121); *lpr* (p. 1128); *lprq* (p. 1135); *lprrm* (p. 1137); *netstat* (p. 1339); *newgrp* (p. 1346); *passwd* (p. 1434); ; ; ; ; ; ; *ping* (p. 1543); *ping6* (p. 1549); *pppd* (p. 1560); ; qnxinstall; *rcp* (p. 1658); *rlogin* (p. 1669); *rsh* (p. 1703); *su* (p. 1872); *traceroute* (p. 1985); *traceroute6* (p. 1991).

Errata

devf-* **(p. 169)**

We've corrected the description of the -r option for the devf-* drivers. You should always specify this option, unless you're debugging a problem concerning filesystem corruption.

grep **(p. 914)**

Only grep supports the -h option (which is a Neutrino extension); egrep and fgrep don't.

mkefs **(p. 1209),** ***mketfs*** **(p. 1219),** ***mkifs*** **(p. 1241)**

The description of the perms is now correct; for an inline file, the default permissions are 0666.

show_vesa **(p. 1767)**

You must log in as root and be in text mode to run this utility. This utility doesn't have any options.

stty **(p. 1863)**

The stty under Neutrino doesn't support the lkhflow or lksflow option.

Copyright © 2014, QNX Software Systems Limited

What's new in the QNX Neutrino Core OS 6.3.2?

New entries

aps **(p. 41)**

Manage adaptive scheduler partitions

Changed content

devb-ram **(p. 258)**

Added address, blksize, and nodinit to the list of ram options.

mkifs **(p. 1241)**

Now includes the module attribute for loading optional modules into `procnto`.

on **(p. 1417)**

Now includes -C, -i, and -R options for specifying the runmask for the new process.

pidin **(p. 1521)**

Now includes `H`, `h`, `i`, and `o` formatting codes, and extsched, fds, regs, sched, and threads shorthand names.

procnto **(p. 1586)**

Now includes -m and -c options.

slay **(p. 1774)**

Now includes -C, -i, and -R options for manipulating the runmask, and -T for identifying the thread that you want to send a signal to or whose runmask or priority you want to change. Also includes -m option to restrict the match.

-P without -T now affects all threads in the specified process or processes. Previously it affected only thread 1.

This utility now matches process IDs or names; you can use the new -m option to limit which it matches.

startup-* **(p. 1729)**

Copyright © 2014, QNX Software Systems Limited

Now include a -F option to control the *flags* field in the *cpuinfo* section of the system page.

qconfig **(p. 1618)**

Added the -a, -b, and -p options.

Errata

dcheck **(p. 175)**

Corrected the name of the `.bad_blks` file.

passwd **(p. 1434)**

The second field of entries in the `/etc/passwd` file is an `x`. It represents the group password, which Neutrino doesn't support.

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Momentics 6.3.0 Service Pack 2?

New entries

addr2line **(p. 34)**

Convert addresses into line number/file name pairs

c++filt **(p. 84)**

Demangle C++ and Java symbols

g++ **(p. 862)**

Compile a program

mq **(p. 1317)**

A new server to manage POSIX message queues. This is an alternate implementation that uses asynchronous messages

ranlib **(p. 1657)**

Index an archive

Changed content

devu-kbd

Reinstated class driver for USB keyboards

devu-mouse

Reinstated class driver for USB mice

esh **(p. 705),** ***fesh*** **(p. 742)**

Added the `emount` and `ewaitfor` builtin commands.

nto*-* variants

Added synopsis information for the `ntoarm-*`, `ntomulti-*`, `ntox86-*` variants of: *ar* (p. 45), *gcc* (p. 889), *gcov* (p. 890), *gdb* (p. 892), *gprof* (p. 913), *ld* (p. 1096), *nm* (p. 1358), *objcopy* (p. 1406), *objdump* (p. 1407), *size* (p. 1771), *strings* (p. 1861), and *strip* (p. 1862).

qcc **(p. 1608)**

Copyright © 2014, QNX Software Systems Limited

The -w9 option's behavior is specifically documented, with additional options necessary to report *all* warnings.

qconn **(p. 1621)**

Added more description to the qconn_prio= option to prevent errors when the system is under heavy load.

sendnto **(p. 1741)**

Documented -b option.

su **(p. 1872)**

Documented - *userid* argument.

tracelogger **(p. 1974)**

Now accepts control commands through its device, *devctl()* and pulses, and has a new option (-c).

uesh **(p. 1998)**

Added the `emount` and `ewaitfor` builtin commands.

usemsg **(p. 2030)**

`%1>` and `%2>` macros fixed (were previously documented as `%0>` and `%1>`).

`usemsg` also uses *`objcopy`* (p. 1406) by default instead of *`ldrel`* (p. 1098).

Errata

io-blk.so **(p. 993)**

If you don't specify a full path for the device in the automount option, `io-blk.so` uses the value of its devdir option as a prefix.

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Momentics 6.3.0 Service Pack 1?

New entries

dumpefs **(p. 651)**

Dump an embedded filesystem

gcov **(p. 890)**

Gather code coverage data

gprof **(p. 913)**

Code profiler

lsm-sctp.so

SCTP service module

qde

Launch the QNX development environment

unzip **(p. 2019)**

Extract zip files

zip **(p. 2081)**

Compress and package files

Changed content

bootpd **(p. 65)**

Now included in the Neutrino runtime.

devb-ncr8

Added the did option.

devf-* **(p. 169)**

Removed references to ffs2 and removed *mountpoint*`/.cmp` from all the `devf-*` drivers.

dhcp.client **(p. 544)**

Copyright © 2014, QNX Software Systems Limited

Added the -T option and a description of the */etc/dhcp/dhcp-options* (p. 550) file.

getconf **(p. 897)**

Added _CS_LOCALE to the list of configuration strings.

io-net

The /dev/io-net directory doesn't appear until a driver or protocol module adds an entry to it.

npm-qnet-l4_lite.so

Qnet is designed to work using its default settings; don't use the options to npm-qnet-l4_lite.so unless you have a specific problem with your environment.

pppd **(p. 1560)**

Added a caveat about spawning pppd with the nodetach or updetach option.

setconf **(p. 1745)**

Corrected the list of configuration strings.

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Momentics 6.3.0?

New entries

devb-adpu320 **(p. 217)**

Adaptec 7901/7902-based SCSI adapters

devb-umass **(p. 262)**

Driver for USB mass storage interface

devc-hspi

Serial driver for Renesas protocol interface

devc-ser8250-ixp2400

Serial driver for Intel IXP425 BSP

devc-sergt64260

Serial driver for Artesyn PM/PPC 750FX BSP

devf-brh

Driver for ADI BRH BSP

devf-ixdp425

Driver for Intel XScale IXDP425 BSP

devn-dm9102.so **(p. 348)**

Driver for Davicom DM9102 Ethernet controllers

devn-tigon3.so

Driver for Broadcom BCM570X Ethernet controllers

devu-ehci.so **(p. 459)**

Driver for Enhanced Host Controller Interface (EHCI) for USB 2.0

devu-ohci.so **(p. 464)**

Driver for Open Host Controller Interface (OHCI) for USB 2.0

devu-uhci.so **(p. 468)**

Driver for Universal Host Controller Interface (UHCI) for USB 2.0

Copyright © 2014, QNX Software Systems Limited

fontview

Font viewer utility

gns **(p. 906)**

Global Name Service Manager

hogs **(p. 939)**

List the processes that are hogging the CPU

icc

Intel C and C++ compiler

io-usb **(p. 1015)**

Server for Universal Serial Bus (USB)

ipf

Alter packet filtering lists

ipfs

Save and restore information for NAT and stat tables

ipfstat

Report on packet filter statistics and filter list

ipmon

Monitor `/dev/ipl` for logged packets

ipnat

User interface to NAT

lsm-ipfilter-*.so

Provide IP filter services

mksbp

Build a QNX System Builder project

mmplay

Multimedia player

netfront

NetFront web server

nfm-autoip.so

Copyright © 2014, QNX Software Systems Limited

AutoIP negotiation module for link-local address

npm-qnet-compat.so

Native QNX Neutrino network manager — compatible version

npm-qnet-l4_lite.so

Lightweight version of native QNX Neutrino network manager

npm-tcpip-v4.so

The original full TCP/IP stack (QNX Neutrino)

npm-tcpip-v6.so

The full TCP/IP stack for IPv6 packets (QNX Neutrino)

ntpd **(p. 1373)**

Network Time Protocol (NTP) daemon

ntpdate **(p. 1379)**

Set the local time and date by polling NTP servers

ntpdc **(p. 1382)**

Query the NTP daemon

ntpq **(p. 1390)**

Monitor NTP daemon and determine its performance

ntptrace **(p. 1403)**

Trace a chain of NTP servers

od **(p. 1408)**

Dump a file in various formats (POSIX)

omshell **(p. 1412)**

Connect, query and change ISC DHCP server's state

pci-brh

PCI controller for ADI BRH BSP

pci-ixc1100

PCI controller for IXDP425 BSP

pci-ixp2400

Copyright © 2014, QNX Software Systems Limited

PCI controller for IXP2400 BSP

pci-ppc440rb

PCI controller for PPC 440GP BSP

playaudio

Play an audio stream

qconfig **(p. 1618)**

Query and display QNX installations and configurations

qnxplayer

CD and audio file player

QWinCfg

Query and display QNX installations and configurations on Windows

rpcbind **(p. 1695)**

Map RPC program numbers into universal addresses

rpcgen **(p. 1697)**

An RPC protocol compiler

script **(p. 1731)**

Create a typescript of a terminal session

startup-artesyn750fx

Startup for the Artesyn PM/PPC 750FX evaluation board

uud **(p. 2035)**

Decode a file that was encoded with uue

uue **(p. 2037)**

Encode a binary file into ASCII

Deleted entries

cl-installer

Use `qnxinstall` instead.

devu-kbd

Copyright © 2014, QNX Software Systems Limited

Class driver for USB keyboards

devu-mouse

Class driver for USB mice

devu-ohci

USB manager for OHCI controllers

devu-uhci

USB managers for UHCI controllers

phflash

Shockwave Flash player

plaympegaudio_noph

Deprecated; replaced by `playaudio`

playsound_noph

Deprecated; replaced by `playaudio`

portmap

Deprecated; replaced by *rpcbind* (p. 1695) utility.

startup-sc400

Deprecated; replaced by *startup-bios* (p. 1858).

devg-chips-hiqv.so

This driver is deprecated.

devg-igs5300.so

This driver is deprecated.

Changed content

devp-pccard **(p. 455)**

Added new options to the documentation.

diff **(p. 619)**

Updated the documentation to reflect the GNU version of `diff` instead of the POSIX version.

dumper **(p. 652)**

Copyright © 2014, QNX Software Systems Limited

Added the -s and -w options.

fontadmin

Added description of the new font options schema.

io-net

Added option and drivers.

mkdir **(p. 1202)**

Removed the -s option.

on **(p. 1417)**

The -p option now supports sporadic scheduling.

qnxinstall

New options and functionality.

sin

The output of the register command now includes the instruction pointer as the first register.

Errata

ksh **(p. 1029)**

Deleted the -F option.

Copyright © 2014, QNX Software Systems Limited

What's new in QNX Momentics 6.2.1?

New entries

addvariant **(p. 35)**

Add a new OS/CPU/VARIANT directory structure to a source tree

devc-amctap

Serial communications manager for AMC WireTAP/PowerTAP

devc_amctap_host

Windows host server for `devc-amctap`

devf-bigsur

SH4 7751 Big Sur eval board

devg-pxa250.so

Graphics driver for Intel PXA250 LCD controller

devh-usb.so **(p. 337)**

Driver for USB-compliant human-interface devices (HID)

devh-ps2ser.so **(p. 331)**

Driver for serial and PS2 human-interface devices (HID)

devi-hid **(p. 339)**

Universal input manager for keyboards and mice

devn-i82544.so **(p. 364)**

Network driver for Intel Gigabit Ethernet LAN adapters

devn-ppc405.so

Network driver for IBM PPC405 on-chip Ethernet controller

devn-prism.so

Network driver for PRISM-based wireless Ethernet controller

hidview **(p. 937)**

Display HID device information

Copyright © 2014, QNX Software Systems Limited

io-hid **(p. 1005)**

Start a manager for HID input devices

qnxinstall

GUI-based QNX Software Installer (QSI)

startup-aspen

Startup the Renesas Aspen evaluation board (SH)

waitfor **(p. 2044)**

Wait until a path exists

Deleted entries

ci

Check in RCS revisions (UNIX)

co

Check out RCS revisions (UNIX)

pkg-installer

Use `qnxinstall` instead.

rcs

Change RCS file attributes (UNIX)

rcsclean

Clean up working files (UNIX)

rcsdiff

Compare RCS revisions (UNIX)

rcsmerge

Merge RCS revisions (UNIX)

Changed content

devb-eide **(p. 235)**

Added the lba48 option.

devb-ncr8

Copyright © 2014, QNX Software Systems Limited

Added the nosync and nowide options.

devn-* **(p. 169)**

The verbose option has been standardized and its output now goes to `slogger` (view using `sloginfo`).

diskboot

Removed -f option

dumper **(p. 652)**

Clarified home directory behavior

flashctl **(p. 771)**

Added example to clarify partition organization.

login **(p. 1121)**

Clarified the behavior of the -f option

mkifs **(p. 1241)**

Clarified the `type` note and added the `dperms` attribute

pidin **(p. 1521)**

Added the irqs, net, rc, and timers shorthand names.

ping **(p. 1543)**

Added many new options

qconn **(p. 1621)**

Added *_prio options

sendnto **(p. 1741)**

General updates, clarifications and corrections

sin

Explained side-channels

startup-* **(p. 1729)**

Added the -x option for x86 boards.

telnet **(p. 1931)**

Removed -a, -l, and -K options

Copyright © 2014, QNX Software Systems Limited

Glossary

A20 gate

On x86-based systems, a hardware component that forces the A20 address line on the bus to zero, regardless of the actual setting of the A20 address line on the processor. This component is in place to support legacy systems, but the QNX Neutrino RTOS doesn't require any such hardware. Note that some processors, such as the 386EX, have the A20 gate hardware built right into the processor itself — our IPL will disable the A20 gate as soon as possible after startup.

adaptive

Scheduling policy whereby a thread's priority is decayed by 1. See also *FIFO*, *round robin*, and *sporadic*.

adaptive partitioning

A method of dividing, in a flexible manner, CPU time, memory, file resources, or kernel resources with some policy of minimum guaranteed usage.

application ID

A number that identifies all processes that are part of an application. Like process group IDs, the application ID value is the same as the process id of the first process in the application. A new application is created by spawning with the `POSIX_SPAWN_NEWAPP` or `SPAWN_NEWAPP` flag. A process created without one of those inherits the application ID of its parent. A process needs the `PROCMGR_AID_CHILD_NEWAPP` ability in order to set those flags.

The *SignalKill()* kernel call accepts a `SIG_APPID` flag ORed into the signal number parameter. This tells it to send the signal to all the processes with an application ID that matches the *pid* argument. The `DCMD_PROC_INFO` *devctl()* returns the application ID in a structure field.

asymmetric multiprocessing (AMP)

A multiprocessing system where a separate OS, or a separate instantiation of the same OS, runs on each CPU.

atomic

Of or relating to atoms. :-)

In operating systems, this refers to the requirement that an operation, or sequence of operations, be considered *indivisible*. For example, a thread may need to move a file position to a given location and read data. These operations must be performed in an atomic manner; otherwise, another

Copyright © 2014, QNX Software Systems Limited

thread could preempt the original thread and move the file position to a different location, thus causing the original thread to read data from the second thread's position.

attributes structure

Structure containing information used on a per-resource basis (as opposed to the *OCB*, which is used on a per-open basis).

This structure is also known as a *handle*. The structure definition is fixed (`iofunc_attr_t`), but may be extended. See also *mount structure*.

bank-switched

A term indicating that a certain memory component (usually the device holding an *image*) isn't entirely addressable by the processor. In this case, a hardware component manifests a small portion (or "window") of the device onto the processor's address bus. Special commands have to be issued to the hardware to move the window to different locations in the device. See also *linearly mapped*.

base layer calls

Convenient set of library calls for writing resource managers. These calls all start with *resmgr_*()*. Note that while some base layer calls are unavoidable (e.g. *resmgr_pathname_attach()*), we recommend that you use the *POSIX layer calls* where possible.

BIOS/ROM Monitor extension signature

A certain sequence of bytes indicating to the BIOS or ROM Monitor that the device is to be considered an "extension" to the BIOS or ROM Monitor — control is to be transferred to the device by the BIOS or ROM Monitor, with the expectation that the device will perform additional initializations.

On the x86 architecture, the two bytes `0x55` and `0xAA` must be present (in that order) as the first two bytes in the device, with control being transferred to offset `0x0003`.

block-integral

The requirement that data be transferred such that individual structure components are transferred in their entirety — no partial structure component transfers are allowed.

In a resource manager, directory data must be returned to a client as *block-integral* data. This means that only complete `struct dirent` structures can be returned — it's inappropriate to return partial structures,

Copyright © 2014, QNX Software Systems Limited

assuming that the next `_IO_READ` request will "pick up" where the previous one left off.

bootable

An image can be either bootable or *nonbootable*. A bootable image is one that contains the startup code that the IPL can transfer control to.

bootfile

The part of an OS image that runs the *startup code* and the microkernel.

bound multiprocessing (BMP)

A multiprocessing system where a single instantiation of an OS manages all CPUs simultaneously, but you can lock individual applications or threads to a specific CPU.

budget

In *sporadic* scheduling, the amount of time a thread is permitted to execute at its normal priority before being dropped to its low priority.

buildfile

A text file containing instructions for `mkifs` specifying the contents and other details of an *image*, or for `mkefs` specifying the contents and other details of an embedded filesystem image.

canonical mode

Also called edited mode or "cooked" mode. In this mode the character device library performs line-editing operations on each received character. Only when a line is "completely entered" — typically when a carriage return (CR) is received — will the line of data be made available to application processes. Contrast *raw mode*.

channel

A kernel object used with message passing.

In QNX Neutrino, message passing is directed towards a *connection* (made to a channel); threads can receive messages from channels. A thread that wishes to receive messages creates a channel (using *ChannelCreate()*), and then receives messages from that channel (using *MsgReceive()*). Another thread that wishes to send a message to the first thread must make a connection to that channel by "attaching" to the channel (using *ConnectAttach()*) and then sending data (using *MsgSend()*).

chid

Copyright © 2014, QNX Software Systems Limited

An abbreviation for *channel ID*.

CIFS

Common Internet File System (also known as SMB) — a protocol that allows a client workstation to perform transparent file access over a network to a Windows 95/98/NT server. Client file access calls are converted to CIFS protocol requests and are sent to the server over the network. The server receives the request, performs the actual filesystem operation, and sends a response back to the client.

CIS

Card Information Structure — a data block that maintains information about flash configuration. The CIS description includes the types of memory devices in the regions, the physical geometry of these devices, and the partitions located on the flash.

coid

An abbreviation for *connection ID*.

combine message

A resource manager message that consists of two or more messages. The messages are constructed as combine messages by the client's C library (e.g. *stat()*, *readblock()*), and then handled as individual messages by the resource manager.

The purpose of combine messages is to conserve network bandwidth and/or to provide support for atomic operations. See also *connect message* and *I/O message*.

connect message

In a resource manager, a message issued by the client to perform an operation based on a pathname (e.g. an `io_open` message). Depending on the type of connect message sent, a context block (see *OCB*) may be associated with the request and will be passed to subsequent I/O messages. See also *combine message* and *I/O message*.

connection

A kernel object used with message passing.

Connections are created by client threads to “connect” to the channels made available by servers. Once connections are established, clients can *MsgSendv()* messages over them. If a number of threads in a process all attach to the same channel, then the one connection is shared among all

Copyright © 2014, QNX Software Systems Limited

the threads. Channels and connections are identified within a process by a small integer.

The key thing to note is that connections and file descriptors (*FD*) are one and the same object. See also *channel* and *FD*.

context

Information retained between invocations of functionality.

When using a resource manager, the client sets up an association or *context* within the resource manager by issuing an *open()* call and getting back a file descriptor. The resource manager is responsible for storing the information required by the context (see *OCB*). When the client issues further file-descriptor based messages, the resource manager uses the OCB to determine the context for interpretation of the client's messages.

cooked mode

See *canonical mode*.

core dump

A file describing the state of a process that terminated abnormally.

critical section

A code passage that *must* be executed “serially” (i.e. by only one thread at a time). The simplest from of critical section enforcement is via a *mutex*.

deadlock

A condition in which one or more threads are unable to continue due to resource contention. A common form of deadlock can occur when one thread sends a message to another, while the other thread sends a message to the first. Both threads are now waiting for each other to reply to the message. Deadlock can be avoided by good design practices or massive kludges — we recommend the good design approach.

device driver

A process that allows the OS and application programs to make use of the underlying hardware in a generic way (e.g. a disk drive, a network interface). Unlike OSs that require device drivers to be tightly bound into the OS itself, device drivers for the QNX Neutrino RTOS are standard processes that can be started and stopped dynamically. As a result, adding device drivers doesn't affect any other part of the OS — drivers can be developed and debugged like any other application. Also, device drivers are in their own protected address space, so a bug in a device driver won't cause the entire OS to shut down.

Copyright © 2014, QNX Software Systems Limited

discrete (or traditional) multiprocessor system

A system that has separate physical processors hooked up in multiprocessing mode over a board-level bus.

DNS

Domain Name Service — an Internet protocol used to convert ASCII domain names into IP addresses. In QNX Neutrino native networking, `dns` is one of *Qnet*'s builtin resolvers.

dynamic bootfile

An OS image built on the fly. Contrast *static bootfile*.

dynamic linking

The process whereby you link your modules in such a way that the Process Manager will link them to the library modules before your program runs. The word "dynamic" here means that the association between your program and the library modules that it uses is done *at load time*, not at linktime. Contrast *static linking*. See also *runtime loading*.

edge-sensitive

One of two ways in which a *PIC* (Programmable Interrupt Controller) can be programmed to respond to interrupts. In edge-sensitive mode, the interrupt is "noticed" upon a transition to/from the rising/falling edge of a pulse. Contrast *level-sensitive*.

edited mode

See *canonical mode*.

EOI

End Of Interrupt — a command that the OS sends to the PIC after processing all Interrupt Service Routines (ISR) for that particular interrupt source so that the PIC can reset the processor's In Service Register. See also *PIC* and *ISR*.

EPROM

Erasable Programmable Read-Only Memory — a memory technology that allows the device to be programmed (typically with higher-than-operating voltages, e.g. 12V), with the characteristic that any bit (or bits) may be individually programmed from a `1` state to a `0` state. To change a bit from a `0` state into a `1` state can only be accomplished by erasing the *entire* device, setting *all* of the bits to a `1` state. Erasing is accomplished by shining an ultraviolet light through the erase window of the device for a fixed period

Copyright © 2014, QNX Software Systems Limited

of time (typically 10-20 minutes). The device is further characterized by having a limited number of erase cycles (typically 10e5 - 10e6). Contrast *flash* and *RAM*.

event

A notification scheme used to inform a thread that a particular condition has occurred. Events can be signals or pulses in the general case; they can also be unblocking events or interrupt events in the case of kernel timeouts and interrupt service routines. An event is delivered by a thread, a timer, the kernel, or an interrupt service routine when appropriate to the requestor of the event.

FD

File Descriptor — a client must open a file descriptor to a resource manager via the *open()* function call. The file descriptor then serves as a handle for the client to use in subsequent messages. Note that a file descriptor is the exact same object as a connection ID (*coid*, returned by *ConnectAttach()*).

FIFO

First In First Out — a scheduling policy whereby a thread is able to consume CPU at its priority level without bounds. See also *adaptive*, *round robin*, and *sporadic*.

flash memory

A memory technology similar in characteristics to *EPROM* memory, with the exception that erasing is performed electrically instead of via ultraviolet light, and, depending upon the organization of the flash memory device, erasing may be accomplished in blocks (typically 64 KB at a time) instead of the entire device. Contrast *EPROM* and *RAM*.

FQNN

Fully Qualified Node Name — a unique name that identifies a QNX Neutrino node on a network. The FQNN consists of the nodename plus the node domain tacked together.

garbage collection

Also known as space reclamation, the process whereby a filesystem manager recovers the space occupied by deleted files and directories.

HA

High Availability — in telecommunications and other industries, HA describes a system's ability to remain up and running without interruption for extended periods of time.

Copyright © 2014, QNX Software Systems Limited

handle

A pointer that the resource manager base library binds to the pathname registered via *resmgr_attach()*. This handle is typically used to associate some kind of per-device information. Note that if you use the *iofunc_*()* *POSIX layer calls*, you must use a particular *type* of handle — in this case called an *attributes structure*.

hard thread affinity

A user-specified binding of a thread to a set of processors, done by means of a *runmask*. Contrast *soft thread affinity*.

image

In the context of embedded QNX Neutrino systems, an "image" can mean either a structure that contains files (i.e. an OS image) or a structure that can be used in a read-only, read/write, or read/write/reclaim FFS-2-compatible filesystem (i.e. a flash filesystem image).

inherit mask

A bitmask that specifies which processors a thread's children can run on. Contrast *runmask*.

interrupt

An event (usually caused by hardware) that interrupts whatever the processor was doing and asks it do something else. The hardware will generate an interrupt whenever it has reached some state where software intervention is required.

interrupt handler

See *ISR*.

interrupt latency

The amount of elapsed time between the generation of a hardware interrupt and the first instruction executed by the relevant interrupt service routine. Also designated as "T_{il}". Contrast *scheduling latency*.

interrupt service routine

See *ISR*.

interrupt service thread

A thread that is responsible for performing thread-level servicing of an interrupt.

Copyright © 2014, QNX Software Systems Limited

Since an *ISR* can call only a very limited number of functions, and since the amount of time spent in an ISR should be kept to a minimum, generally the bulk of the interrupt servicing work should be done by a thread. The thread attaches the interrupt (via *InterruptAttach()* or *InterruptAttachEvent()*) and then blocks (via *InterruptWait()*), waiting for the ISR to tell it to do something (by returning an event of type `SIGEV_INTR`). To aid in minimizing *scheduling latency*, the interrupt service thread should raise its priority appropriately.

I/O message

A message that relies on an existing binding between the client and the resource manager. For example, an `_IO_READ` message depends on the client's having previously established an association (or *context*) with the resource manager by issuing an *open()* and getting back a file descriptor. See also *connect message*, *context*, *combine message*, and *message*.

I/O privileges

A particular right, that, if enabled for a given thread, allows the thread to perform I/O instructions (such as the x86 assembler `in` and `out` instructions). By default, I/O privileges are disabled, because a program with it enabled can wreak havoc on a system. To enable I/O privileges, the thread must be running as `root`, and call *ThreadCtl()*.

IPC

Interprocess Communication — the ability for two processes (or threads) to communicate. The QNX Neutrino RTOS offers several forms of IPC, most notably native messaging (synchronous, client/server relationship), POSIX message queues and pipes (asynchronous), as well as signals.

IPL

Initial Program Loader — the software component that either takes control at the processor's reset vector (e.g. location `0xFFFFFFF0` on the x86), or is a BIOS extension. This component is responsible for setting up the machine into a usable state, such that the startup program can then perform further initializations. The IPL is written in assembler and C. See also *BIOS extension signature* and *startup code*.

IRQ

Interrupt Request — a hardware request line asserted by a peripheral to indicate that it requires servicing by software. The IRQ is handled by the *PIC*, which then interrupts the processor, usually causing the processor to execute an *Interrupt Service Routine (ISR)*.

Copyright © 2014, QNX Software Systems Limited

ISR

Interrupt Service Routine — a routine responsible for servicing hardware (e.g. reading and/or writing some device ports), for updating some data structures shared between the ISR and the thread(s) running in the application, and for signalling the thread that some kind of event has occurred.

kernel

See *microkernel*.

level-sensitive

One of two ways in which a *PIC* (Programmable Interrupt Controller) can be programmed to respond to interrupts. If the PIC is operating in level-sensitive mode, the IRQ is considered active whenever the corresponding hardware line is active. Contrast *edge-sensitive*.

linearly mapped

A term indicating that a certain memory component is entirely addressable by the processor. Contrast *bank-switched*.

message

A parcel of bytes passed from one process to another. The OS attaches no special meaning to the content of a message — the data in a message has meaning for the sender of the message and for its receiver, but for no one else.

Message passing not only allows processes to pass data to each other, but also provides a means of synchronizing the execution of several processes. As they send, receive, and reply to messages, processes undergo various "changes of state" that affect when, and for how long, they may run.

microkernel

A part of the operating system that provides the minimal services used by a team of optional cooperating processes, which in turn provide the higher-level OS functionality. The microkernel itself lacks filesystems and many other services normally expected of an OS; those services are provided by optional processes.

mount structure

An optional, well-defined data structure (of type `iofunc_mount_t`) within an *iofunc_*()* structure, which contains information used on a per-mountpoint basis (generally used only for filesystem resource managers). See also *attributes structure* and *OCB*.

Copyright © 2014, QNX Software Systems Limited

mountpoint

The location in the pathname space where a resource manager has "registered" itself. For example, the serial port resource manager registers mountpoints for each serial device (`/dev/ser1`, `/dev/ser2`, etc.), and a CD-ROM filesystem may register a single mountpoint of `/cdrom`.

multicore system

A chip that has one physical processor with multiple CPUs interconnected over a chip-level bus.

mutex

Mutual exclusion lock, a simple synchronization service used to ensure exclusive access to data shared between threads. It is typically acquired (*pthread_mutex_lock()*) and released (*pthread_mutex_unlock()*) around the code that accesses the shared data (usually a *critical section*). See also *critical section*.

name resolution

In a QNX Neutrino network, the process by which the *Qnet* network manager converts an *FQNN* to a list of destination addresses that the transport layer knows how to get to.

name resolver

Program code that attempts to convert an *FQNN* to a destination address.

nd

An abbreviation for *node descriptor*, a numerical identifier for a node *relative to the current node*. Each node's node descriptor for itself is 0 (`ND_LOCAL_NODE`).

NDP

Node Discovery Protocol — proprietary QNX Software Systems protocol for broadcasting name resolution requests on a QNX Neutrino LAN.

network directory

A directory in the pathname space that's implemented by the *Qnet* network manager.

NFS

Network FileSystem — a TCP/IP application that lets you graft remote filesystems (or portions of them) onto your local namespace. Directories on the remote systems appear as part of your local filesystem and all the utilities

Copyright © 2014, QNX Software Systems Limited

you use for listing and managing files (e.g. `ls`, `cp`, `mv`) operate on the remote files exactly as they do on your local files.

NMI

Nonmaskable Interrupt — an interrupt that can't be masked by the processor. We don't recommend using an NMI!

Node Discovery Protocol

See *NDP*.

node domain

A character string that the *Qnet* network manager tacks onto the nodename to form an *FQNN*.

nodename

A unique name consisting of a character string that identifies a node on a network.

nonbootable

A nonbootable OS image is usually provided for larger embedded systems or for small embedded systems where a separate, configuration-dependent setup may be required. Think of it as a second “filesystem” that has some additional files on it. Since it's nonbootable, it typically won't contain the OS, startup file, etc. Contrast *bootable*.

OCB

Open Control Block (or Open Context Block) — a block of data established by a resource manager during its handling of the client's *open()* function. This context block is bound by the resource manager to this particular request, and is then automatically passed to all subsequent I/O functions generated by the client on the file descriptor returned by the client's *open()*.

package filesystem

A virtual filesystem manager that presents a customized view of a set of files and directories to a client. The “real” files are present on some medium; the package filesystem presents a virtual view of selected files to the client.

partition

A division of CPU time, memory, file resources, or kernel resources with some policy of minimum guaranteed usage.

pathname prefix

See *mountpoint*.

Copyright © 2014, QNX Software Systems Limited

pathname space mapping

The process whereby the Process Manager maintains an association between resource managers and entries in the pathname space.

persistent

When applied to storage media, the ability for the medium to retain information across a power-cycle. For example, a hard disk is a persistent storage medium, whereas a ramdisk is not, because the data is lost when power is lost.

PIC

Programmable Interrupt Controller — hardware component that handles IRQs. See also *edge-sensitive*, *level-sensitive*, and *ISR*.

PID

Process ID. Also often *pid* (e.g. as an argument in a function call).

POSIX

An IEEE/ISO standard. The term is an acronym (of sorts) for Portable Operating System Interface — the "X" alludes to "UNIX", on which the interface is based.

POSIX layer calls

Convenient set of library calls for writing resource managers. The POSIX layer calls can handle even more of the common-case messages and functions than the *base layer calls*. These calls are identified by the *iofunc_*()* prefix. In order to use these (and we strongly recommend that you do), you must also use the well-defined POSIX-layer *attributes* (`iofunc_attr_t`), *OCB* (`iofunc_ocb_t`), and (optionally) *mount* (`iofunc_mount_t`) structures.

preemption

The act of suspending the execution of one thread and starting (or resuming) another. The suspended thread is said to have been "preempted" by the new thread. Whenever a lower-priority thread is actively consuming the CPU, and a higher-priority thread becomes READY, the lower-priority thread is immediately preempted by the higher-priority thread.

prefix tree

The internal representation used by the Process Manager to store the pathname table.

priority inheritance

Copyright © 2014, QNX Software Systems Limited

The characteristic of a thread that causes its priority to be raised or lowered to that of the thread that sent it a message. Also used with mutexes. Priority inheritance is a method used to prevent *priority inversion*.

priority inversion

A condition that can occur when a low-priority thread consumes CPU at a higher priority than it should. This can be caused by not supporting priority inheritance, such that when the lower-priority thread sends a message to a higher-priority thread, the higher-priority thread consumes CPU *on behalf of* the lower-priority thread. This is solved by having the higher-priority thread inherit the priority of the thread on whose behalf it's working.

process

A nonschedulable entity, which defines the address space and a few data areas. A process must have at least one *thread* running in it — this thread is then called the first thread.

process group

A collection of processes that permits the signalling of related processes. Each process in the system is a member of a process group identified by a process group ID. A newly created process joins the process group of its creator.

process group ID

The unique identifier representing a process group during its lifetime. A process group ID is a positive integer. The system may reuse a process group ID after the process group dies.

process group leader

A process whose ID is the same as its process group ID.

process ID (PID)

The unique identifier representing a process. A PID is a positive integer. The system may reuse a process ID after the process dies, provided no existing process group has the same ID. Only the Process Manager can have a process ID of 1.

pty

Pseudo-TTY — a character-based device that has two “ends”: a master end and a slave end. Data written to the master end shows up on the slave end, and vice versa. These devices are typically used to interface between a program that expects a character device and another program that wishes

Copyright © 2014, QNX Software Systems Limited

to use that device (e.g. the shell and the `telnet` daemon process, used for logging in to a system over the Internet).

pulses

In addition to the synchronous Send/Receive/Reply services, QNX Neutrino also supports fixed-size, nonblocking messages known as pulses. These carry a small payload (four bytes of data plus a single byte code). A pulse is also one form of *event* that can be returned from an ISR or a timer. See *MsgDeliverEvent()* for more information.

Qnet

The native network manager in the QNX Neutrino RTOS.

QoS

Quality of Service — a policy (e.g. `loadbalance`) used to connect nodes in a network in order to ensure highly dependable transmission. QoS is an issue that often arises in high-availability (*HA*) networks as well as realtime control systems.

RAM

Random Access Memory — a memory technology characterized by the ability to read and write any location in the device without limitation. Contrast *flash* and *EPROM*.

raw mode

In raw input mode, the character device library performs no editing on received characters. This reduces the processing done on each character to a minimum and provides the highest performance interface for reading data. Also, raw mode is used with devices that typically generate binary data — you don't want any translations of the raw binary stream between the device and the application. Contrast *canonical mode*.

replenishment

In *sporadic* scheduling, the period of time during which a thread is allowed to consume its execution *budget*.

reset vector

The address at which the processor begins executing instructions after the processor's reset line has been activated. On the x86, for example, this is the address `0xFFFFFFF0`.

resource manager

Copyright © 2014, QNX Software Systems Limited

A user-level server program that accepts messages from other programs and, optionally, communicates with hardware. QNX Neutrino resource managers are responsible for presenting an interface to various types of devices, whether actual (e.g. serial ports, parallel ports, network cards, disk drives) or virtual (e.g. `/dev/null`, a network filesystem, and pseudo-ttys).

In other operating systems, this functionality is traditionally associated with *device drivers*. But unlike device drivers, QNX Neutrino resource managers don't require any special arrangements with the kernel. In fact, a resource manager looks just like any other user-level program. See also *device driver*.

RMA

Rate Monotonic Analysis — a set of methods used to specify, analyze, and predict the timing behavior of realtime systems.

round robin

A scheduling policy whereby a thread is given a certain period of time to run. Should the thread consume CPU for the entire period of its timeslice, the thread will be placed at the end of the ready queue for its priority, and the next available thread will be made READY. If a thread is the only thread READY at its priority level, it will be able to consume CPU again immediately. See also *adaptive*, *FIFO*, and *sporadic*.

runmask

A bitmask that indicates which processors a thread can run on. Contrast *inherit mask*.

runtime loading

The process whereby a program decides *while it's actually running* that it wishes to load a particular function from a library. Contrast *static linking*.

scheduling latency

The amount of time that elapses between the point when one thread makes another thread READY and when the other thread actually gets some CPU time. Note that this latency is almost always at the control of the system designer.

Also designated as "T_{sl}". Contrast *interrupt latency*.

scoid

An abbreviation for *server connection ID*.

session

Copyright © 2014, QNX Software Systems Limited

A collection of process groups established for job control purposes. Each process group is a member of a session. A process belongs to the session that its process group belongs to. A newly created process joins the session of its creator. A process can alter its session membership via *setsid()*. A session can contain multiple process groups.

session leader

A process whose death causes all processes within its process group to receive a SIGHUP signal.

soft thread affinity

The scheme whereby the microkernel tries to dispatch a thread to the processor where it last ran, in an attempt to reduce thread migration from one processor to another, which can affect cache performance. Contrast *hard thread affinity*.

software interrupts

Similar to a hardware interrupt (see *interrupt*), except that the source of the interrupt is software.

sporadic

A scheduling policy whereby a thread's priority can oscillate dynamically between a “foreground” or normal priority and a “background” or low priority. A thread is given an execution *budget* of time to be consumed within a certain *replenishment* period. See also *adaptive*, *FIFO*, and *round robin*.

startup code

The software component that gains control after the IPL code has performed the minimum necessary amount of initialization. After gathering information about the system, the startup code transfers control to the OS.

static bootfile

An image created at one time and then transmitted whenever a node boots. Contrast *dynamic bootfile*.

static linking

The process whereby you combine your modules with the modules from the library to form a single executable that's entirely self-contained. The word “static” implies that it's not going to change — *all* the required modules are already combined into one.

symmetric multiprocessing (SMP)

Copyright © 2014, QNX Software Systems Limited

A multiprocessor system where a single instantiation of an OS manages all CPUs simultaneously, and applications can float to any of them.

system page area

An area in the kernel that is filled by the startup code and contains information about the system (number of bytes of memory, location of serial ports, etc.) This is also called the SYSPAGE area.

thread

The schedulable entity under the QNX Neutrino RTOS. A thread is a flow of execution; it exists within the context of a *process*.

tid

An abbreviation for *thread ID*.

timer

A kernel object used in conjunction with time-based functions. A timer is created via *timer_create()* and armed via *timer_settime()*. A timer can then deliver an *event*, either periodically or on a one-shot basis.

timeslice

A period of time assigned to a *round-robin* or *adaptive* scheduled thread. This period of time is small (on the order of tens of milliseconds); the actual value shouldn't be relied upon by any program (it's considered bad design).

TLB

An abbreviation for *translation look-aside buffer*. To maintain performance, the processor caches frequently used portions of the external memory page tables in the TLB.

TLS

An abbreviation for *thread local storage*.

Copyright © 2014, QNX Software Systems Limited

Index

Copyright © 2014, QNX Software Systems Limited

A

Copyright © 2014, QNX Software Systems Limited

B

Copyright © 2014, QNX Software Systems Limited

C

Copyright © 2014, QNX Software Systems Limited

D

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

E

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

F

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

G

H

Copyright © 2014, QNX Software Systems Limited

I

Copyright © 2014, QNX Software Systems Limited

J

K

Copyright © 2014, QNX Software Systems Limited

L

Copyright © 2014, QNX Software Systems Limited

M

Copyright © 2014, QNX Software Systems Limited

N

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

O

P

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Q

R

Copyright © 2014, QNX Software Systems Limited

S

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited

T

Copyright © 2014, QNX Software Systems Limited

U

Copyright © 2014, QNX Software Systems Limited

V

W

Copyright © 2014, QNX Software Systems Limited

X

Y

Z

Copyright © 2014, QNX Software Systems Limited

Copyright © 2014, QNX Software Systems Limited