Usługi i aplikacje sieciowe

1 Aplikacje sieciowe 2
1.1 Usługi sieciowe 4
1.2 Superserwer sieciowy inetd 5
1.3 Superserwer sieciowy xinetd 8
1.4 Zdalny dostęp do systemu - protokół telnet 13
1.5 Usługa i protokół SSH 17
1.6 Przesyłanie plików, protokół FTP, ... 20
1.7 Protokół SFTP 24
1.8 Protokół SCP 26
2 Sieciowe systemy plików 27
2.1 Informacje wstępne 27
2.2 Działanie 29
2.3 Instalacja i konfiguracja 29
3 System WWW 34
3.1 Architektura WWW 34
3.2 Terminologia: 35
3.3 Język HTML 35
3.4 Identyfikacja strony - adresacja URL 38
3.5 Odsyłacze hipertekstowe pomiędzy dokumentami 38
3.6 Współpraca przeglądarki i serwera WWW 39
3.7 Przeglądarka WWW 39
4 Protokół HTTP 41
4.1 Przeznaczenie 41
4.2 Transakcja HTTP 41
4.3 Przykład transakcji HTTP 43
4.4 Inne metody HTTP 44
4.5 Eksperymenty z HTTP 44
4.6 Kodowanie treści komunikatu 45
5 Literatura 46

1 Aplikacje sieciowe

Sieciowe systemy operacyjne oferują wiele usług sieciowych (jako serwer) a także zawierają liczne narzędzia do korzystania z usług dostępnych w Internecie i oferowanych przez inne systemy.
Podzielić je można na grupy:

- Usługi związane z Internetem
- Zdalne wykonywanie aplikacji
- Sieciowe systemy plików

Funkcja	Protokół	Uwagi
Poczta elektroniczna	pop3, smtp	Klient i serwer poczty
Przesyłanie plików FTP	ftp	Klient i serwer
Trywialne przesyłanie plików	tftp	Wczytywanie systemu przez sieć
Bezpieczne przesyłanie plików	SSH,SFTP	Klient i serwer
Serwer WWW	http	Serwer WWW, przeglądarki
Serwer proxy	http	Przyspiesza dostęp do stron WWW
Pogawędka sieciowa		IRC, ICQ
Telefonia internetowa	VOIP	Skype
Zdalna konfiguracja komputerów	BOOTP, DHCP	Zdalna konfiguracja i wczytywanie systemu
Zdalna autoryzacja	LDAP	
Usługa nazewnicza	DNS	Ustalanie adresu IP na podstawie nazwy komputera i domeny

Tabela 1-1 Usługi związane z Internetem

Funkcja	Uwagi
RPC	Zdalne wykonywanie procedur (Remote Procedure Calls)
telnet	Zdalna konsola, przesyłanie niezaszyfrowanych haseł
ssh,scp	Zdalny dostęp (*Secure Shell*) i kopiowanie plików (*Secure Copy*), usługa zabezpieczona
rlogin, rsh	Zdalne logowanie i shell, usługi niebezpieczne
X– serwer i klient	Zdalna konsola graficzna
VNC	Zdalna konsola graficzna (*Virtua Network Computing*)
rdesktop	Zdalna konsola graficzna dla Windows oparta o protokół RDP (*Remote Desktop Protocol*)

Tabela 1-2 Usługi związane ze zdalnym wykonywaniem aplikacji

NFS	Sieciowy system plików (*Network File System*), klient i serwer usługi
SMB	Dostęp do systemu plików systemu Windows (*Server Message Block*), znany tez jako CIFS (*Common Internet File System)*

Tabela 1-3 Sieciowe systemy plików

1.1 Usługi sieciowe

Plik **`/etc/services`** określa zasób dostępnych usług sieciowych.
Każda linia zawiera kolejno:

- nazwę usługi,
- numer przypisanego jej portu,
- typ usługi
- przypisane danej usłudze aliasy (opcjonalnie).

Przykład pliku **`/etc/services`**:

Usługa	Port/ protokół	Alias	Komentarz
echo	7/tcp		
echo	7/udp		
daytime	13/tcp		
daytime	13/udp		
netstat	15/tcp		
ftp-data	20/tcp		
ftp	21/tcp		
ssh	22/tcp		# SSH Remote Login Protocol
ssh	22/udp		
telnet	23/tcp		
smtp	25/tcp	mail	
time	37/tcp	timserver	
time	37/udp	timserver	
nameserver	42/tcp	name	# IEN 116
tftp	69/udp		
www	80/tcp	http	# WorldWideWeb HTTP
www	80/udp		# HyperText Transfer Protocol
rtelnet	107/tcp		# Remote Telnet
rtelnet	107/udp		
pop3	110/tcp	pop-3	# POP version 3
pop3	110/udp	pop-3	
sunrpc	111/tcp	portmapper	# RPC 4.0 portmapper
sunrpc	111/udp	portmapper	
ntp	123/udp		# Network Time Protocol
snmp	161/tcp		# Simple Net Mgmt Protocol
snmp	161/udp		# Simple Net Mgmt Protocol

Tab. 1-1 Fragment pliku /etc/services

Nr	Nazwa	Funkcja
1	echo	Odsyłanie odebranych znaków do klienta
2	ftp	Przesyłanie plików FTP
3	tftp	Trywialne przesyłanie plików
4	telnet	Interpreter poleceń
5	www	Serwer WWW
6	smtp	Obsługa poczty elektronicznej
7	sunrpc	Łącznik zdalnego wywoływanie procedur RPC
8	pop3	Obsługa poczty elektronicznej
9	rtelnet	Zdalny interpreter poleceń
10	ssh	Bezpieczna konsola
11	netstat	Prezentacja aktualnych połączeń sieciowych

Tabela 1-4 Niektóre usługi wyszczególnione w pliku /etc/services

1.2 Superserwer sieciowy inetd

W systemie Linux i Unix dostępnych jest wiele usług dostępnych przez sieć. Gdyby serwery tych usług były cały czas aktywne blokowałyby zasoby systemowe gdyż znaczna część tych usług byłaby aktywowana tylko niekiedy.

Komputer może świadczyć usługi na dwa sposoby:

- Usługa może być cały czas dostępna – tak jest w przypadku intensywnie wykorzystywanych, uruchamiana jest przy starcie systemu jako demon
- Usługa aktywowana na żądanie – rozwiązanie stosowane gdy usługa potrzebna jest od czasu do czasu.

W praktyce rzadziej potrzebne usługi są aktywowane przez super serwer sieciowy **`inetd`**. W zależności od dystrybucji mogą być używane różne klony programu inetd: xinetd**,** openbsd-inetd, netkit-inetd, inetutils-inetd, micro-inetd, rlinetd .

Plik konfiguracyjny

Przy starcie **`inetd`** odczytuje swój plik konfiguracyjny (zwykle **`/etc/inetd.conf`**).

Każda linia pliku **inetd.conf** zawiera następujące informacje:

- Nazwa usługi
- Styl komunikacji gniazdka – **stream, dgram, raw**.
- Typ protokołu - **udp, tcp**.
- Opcje - **{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]**
- Użytkownik, grupa
- Nazwa programu wykonującego usługę
- Argumenty programu

Opcja **wait|nowait** specyfikuje czy nowy demon usługi ma być uruchomiony współbieżnie czy czekać aż bieżący demon się zakończy. Gniazdka typu **dgram** muszą mieć opcję **wait** podczas gdy gniazda typu **stream** używają zwykle opcji **nowait**.

```
#<service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#discard          stream      tcp   nowait      root  internal
#discard          dgram       udp   wait        root  internal
#daytime          stream      tcp   nowait      root  internal
#time             stream      tcp   nowait      root  internal

#:STANDARD: These are standard services.
ftp    stream tcp nowait root /usr/sbin/tcpd    /usr/sbin/in.ftpd
telnet stream tcp nowait root /usr/ucb/telnetd  in.telnetd
tftp   dgram  udp wait   bin  /usr/ucb/tftpd    in.tftpd
```

Przykład 1-1 Plik konfiguracyjny **/etc/inetd.conf**

Działanie superserwera **inetd**

Na podstawie danych znalezionych w pliku konfiguracyjnym **inetd.conf** tworzy odpowiednią ilość nasłuchujących gniazd - po jednym dla każdej zdefiniowanej tam usługi. Następnie wykonuje **select()** na gniazdach czekając na połączenia przychodzące. Kiedy wykryje, próbę połączenia wywołuje funkcję **accept()** i tworzy nowy proces (**fork()**) obsługujący to nowe połączenie. W kolejnych krokach przekierowuje stdin / stdout do utworzonego gniazda (funkcja **dup2()**) oraz uruchamia właściwego demona obsługującego daną usługę poprzez wykonanie funkcji **exec()**.

Demony realizujące daną usługę mają postać zwykłych programów korzystających ze standardowych strumieni I/O.

Funkcje związane z obsługą sieci są przenoszone na **inetd**. Dzięki temu nasz demon może kontaktować się ze zdalnym klientem

identycznie, jak z użytkownikiem operującym na lokalnej konsoli. Demon może pisać na stdout (np. **printf()**) i czytać ze stdin (**scanf(), read()**). Mechanizm ten zapewnia możliwość sieciowej komunikacji programom, które nie zawierają ani jednej linijki kodu sieciowego.

Poniżej podano przykład takiego programu który zapisuje przychodzące dane do pliku którego nazwa jest pierwszym argumentem.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
  const char *fn = argv[1];
  FILE *fp = fopen(fn, "a+");
  if(fp == NULL) exit(EXIT_FAILURE);
  char str[4096];
 //inetd przesyła informacje do stdin programu
  while(fgets(str, sizeof(str), stdin)) {
    fputs(str, fp);
    fflush(fp);
  }
  fclose(fp);
  return 0;
}
```

Przykład 1-2 Demon sieciowy rejestr zapisujący przychodzące dane do pliku

W pliku /**etc/services** powinien się znaleźć wpis definiujący tę usługę którą możemy nazwać rejestr i będzie zainstalowana na porcie UDP 9999.

rejestr 9999/udp

W pliku **/etc/inetd.conf** należy dodać wpis:

rejestr dgram udp wait root /home/juka/logd /tmp/logfile.txt

Wpis ten informuje że usługa rejestr polega na uruchomieniu programu **logd** który będzie zapisywał dane w pliku **/tmp/logfile.txt**

1.3 Superserwer sieciowy xinetd

Nowszą wersją serwera sieciowego inetd jest xinetd. Zapewnia on większy stopień kontroli nad wykonywanymi aplikacjami. Jest stosowany w dystrybucji Linux Debian i innych.

Uruchomienie

Program **xinetd** uruchamiany jest automatycznie przy starcie systemu poprzez skrypt:

/etc/init.d/xinetd

Skrypt ten ma opcje: **start, stop, reload, restart**

Sam program umieszczony jest w katalogu **/usr/sbin** i uruchamia się:

/usr/sbin/xinetd [opcje]

Konfiguracja

Głównym plikiem konfiguracyjnym superserwera xinetd jest plik : **/etc/xinetd.conf** . Plik ten zazwyczaj zawiera wpis:

includedir /etc/xinet.d

informujący że pliki definiujące poszczególne usługi znajdują się w katalogu: **/etc/xinet.d**

```
$ls /etc/xinetd.d
chargen daytime discard echo time zapisz telnet ftp
```

Każda usługa zdefiniowana jest za pomocą rekordu postaci:

```
service <service_name>
{
        <attribute> <assign_op> <value> <value> ...
        ...
}
```

Definiuje się też parametru domyślne:

```
defaults
{
        <attribute> = <value> <value> ...
        ...
}
```

service_name	Nazwa usługi
attribute	nazwa atrybutu
assign_op	Przypisanie atrybutu znak =, +=, -= = ustawienie atrybutu += dodanie do zbioru atrybutów - = usunięcie ze zbioru atrybutów
value	Wartość atrybutu

Tabela 1-5 Format rekordu definicji usługi

socket_type	Typ gniazdka: stream, dgram, raw
protocol	Typ protokołu z pliku /etc/protocols
wait	Typ obsługi jedno lub wielowątkowy. Gdy wait obsługa następnego zlecenia możliwa po zakończeniu bieżącej. Gdy nowait – obsługa współbieżna
instances	Liczba serwerów współbieżnych dla danej usługi
user	Identyfikator użytkownika procesu obsługi
server	Ścieżka do programu obsługi
server_args	Argumenty programu obsługi
log_type	Typ logu: **SYSLOG syslog_facility [syslog_level]** **FILE file [soft_limit [hard_limit]]**
log_on_success	Co zapisać do logu gdy się uda: PID, HOST, USERID,DURATION
log_on_failure	Co zapisać do logu gdy się nie uda: PID, HOST, USERID,DURATION
port	Określa numer portu dla tej usługi. Musi być zgodne z /etc/services
cps	Liczba połączeń na sek. Składa się z dwóch liczb: liczba połączeń na sekundę i liczba sekund do ponownego dozwolenia usługi.
rlimit_files	Maksymalna liczba otwartych plików dla uslugi

Tabela 1-6 Opcje rekordu definicji usługi

```
service ftp
{
  flags = REUSE
  socket_type = stream
  instances = 50
  wait = no
  user = root
  server = /usr/sbin/in.ftpd
  log_on_success = HOST PID
  log_on_failure = HOST RECORD
}
```

Przykład 1-1 Przykład definicji usługi ftp

Sprawdzenie czy proces się wykonuje

```
$ps –ef | grep xinetd
root 9996 1 0 18:04 ? 00:00:00 /usr/sbin/xinetd
-pidfile /var/run/xinetd.pid -stayalive -inetd_compat
-inetd_ipv6
```

Przykład 1-2 sprawdzenie czy demon xinetd się wykonuje

invoke-rc.c

Instalacja własnej usługi w xinetd

Obecnie pokazane zostanie jak zainstalować własną usługę o nazwie zapisz. Usługa polega na przejęciu linii ze stdin i wpisaniu jej do pliku.

1. Przechodzimy do katalogu roboczego /home/juka/prog/lab/sieci. Tworzymy tam edytorem skrypt w bash o nazwie zapis.sh

```
#!/bin/bash
read -n 512 linia
echo "$linia" >> /home/juka/prog/lab/sieci/zapis.txt
echo Zapisano plik
```

Przykład 1-3 Skrypt **zapis.sh**

Skrypt powoduje odczytanie linii ze **stdin** i dopisanie do pliku **zapis.txt**. Po utworzeniu nadajemy plikowi **zapis.sh** atrybut wykonywalności poprzez polecenie: **chmod a+x zapis.sh** Następnie możemy wypróbować działanie skryptu pisząc: **./zapis.sh**

2. Tworzymy plik definicji usługi o nazwie **zapisz** o zawartości jak w Przykład 1-4. Następnie przełączamy się w tryb root i kopiujemy plik zapisz do katalogu /etc/xinetd.d

```
service zapisz
 {
    type = UNLISTED
    socket_type = stream
    protocol = tcp
    port = 7000
    wait = no
    user = juka
    server = /home/juka/prog/lab/sieci/zapis.sh
    instances = 4
}
```

Przykład 1-4 Plik /etc/xinetd.d/zapisz

```
#ls –l /etc/xinetd.d
-rw-r--r-- 1 root root 798 2008-03-26  chargen
-rw-r--r-- 1 root root 660 2008-03-26  daytime
-rw-r--r-- 1 root root 549 2008-03-26  discard
-rw-r--r-- 1 root root 580 2008-03-26  echo
-rw-r--r-- 1 root root 727 2008-03-26  time
-rw-r--r-- 1 juka juka 196 05-30 18:51 zapisz
```

Przykład 1-5 Zawartość katalogu /etc/xinetd.d

3. Restartujemy demona **xinetd** poprzez polecenie:

 # invoke-rc.d xinetd restart

```
root@maria:/etc/xinetd.d# ls
chargen  daytime  discard  echo  time  zapisz
root@maria:/etc/xinetd.d# invoke-rc.d xinetd restart
Stopping internet superserver: xinetd.
Starting internet superserver: xinetd.
```

Ekran 1-1 Restart demona xinetd

Prawidłowość działania demona xinetd można sprawdzić wyświetlając zawartość pliku /var/log/syslog

4. Za pomocą programu telnet łączymy się z usługą zapisz. Będzie na komputerze lokalnym i porcie 7000.

$telnet 127.0.0.1 7000

Następnie wpisujemy dowolną linie i wciskamy Enter

```
juka@maria:~/prog/lab/sieci$ cat zapis.txt
pierwsza linia
juka@maria:~/prog/lab/sieci$ telnet 127.0.0.1 7000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Ta linia sie dopisze
Zapisano plik
Connection closed by foreign host.
juka@maria:~/prog/lab/sieci$ cat zapis.txt
pierwsza linia
Ta linia sie dopisze
juka@maria:~/prog/lab/sieci$
```

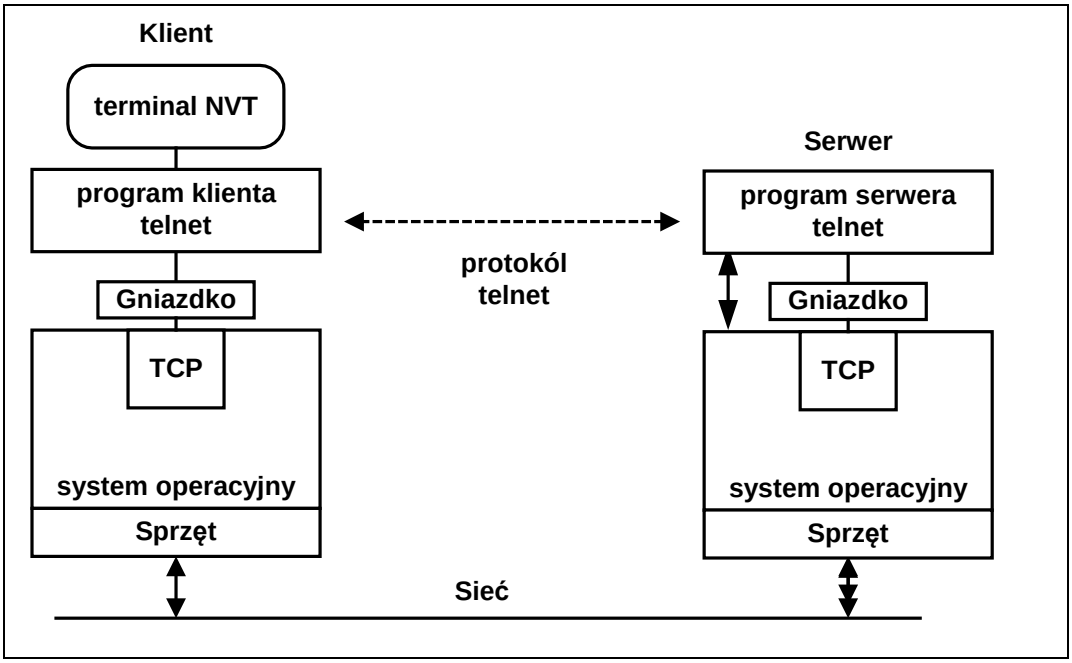
Ekran 1-2 Łączenie się z usługą zapisz za pomocą telnet

1.4 Zdalny dostęp do systemu - protokół telnet

Telnet jest protokołem sieciowym używanym w Internecie i sieciach lokalnych. Protokół telnet jest jednym z najstarszych protokołów sieciowych, opracowany został w 1969 i opisany w RFC 15 i RFC 854. Zapewnia on komunikację ze zdalnym systemem poprzez wirtualny terminal implementowany przez program klienta.

W zamierzeniu telnet ma umożliwiać komunikację pomiędzy różnymi terminalami i systemami operacyjnymi. Terminale posiadają różne właściwości i znaki sterujące. Podobnie z programami obsługującymi terminale po stronie komputera. Protokół telnet oparty jest na następujących zasadach:

- Koncepcji wirtualnego terminala sieciowego NVT (ang. Network Virtual Terminal)
- Procedurze negocjacji opcji pomiędzy terminalem NVT a obsługującym go procesem po stronie zdalnego komputera

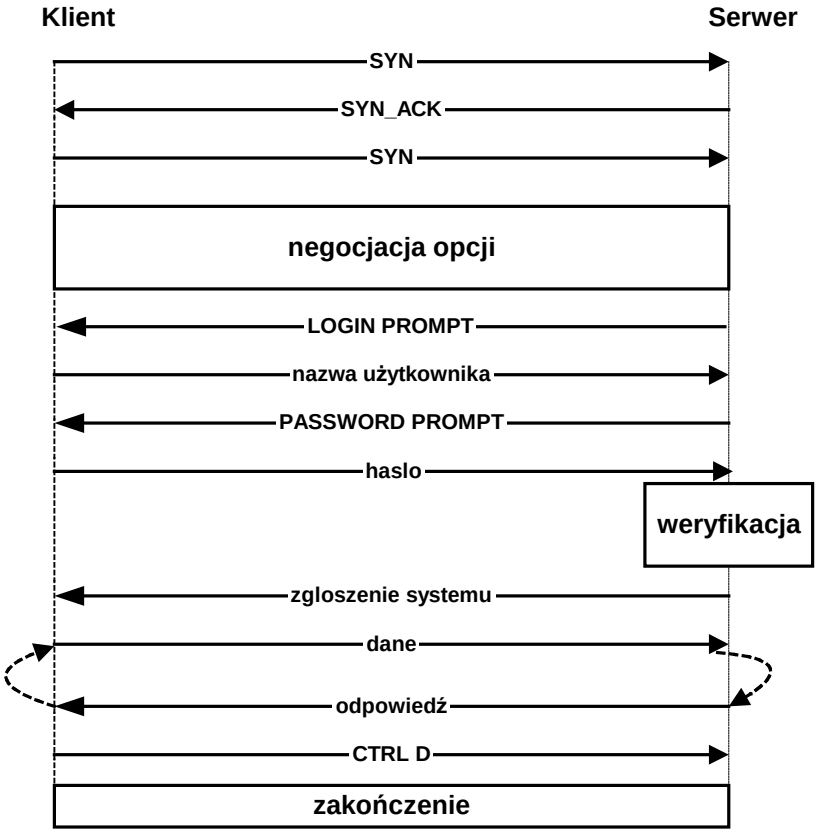


Rys. 1-1 Koncepcja dostępu do zdalnego systemu poprzez protokół telnet

Komunikacja jest 8 bitowa, dwukierunkowa realizowany w warstwie TCP. Przesyłane znaki są kodowane 7 bitowo, pozostałe znaki używane są jako sterujące.

Współpraca:

1. Program klienta, najczęściej o nazwie **`telnet`** sygnalizuje serwerowi chęć nawiązania połączenia. Odbywa się to poprzez wysłanie do serwera polecenia SYN na co ten odpowiada poprzez SYN_ACK co klient potwierdza znakiem ACK.
2. Po tym etapie następuje negocjacja opcji pomiędzy klientem a serwerem.
3. Serwer wysyła zapytanie o nazwę użytkownika i hasło. Gdy weryfikacja przebiegnie pomyślnie, w programie klienta wyświetlany jest znak zachęty i może on wprowadzać polecenia w postaci linii tekstu który przesyłany jest do serwera.
4. Serwer interpretuje polecenia i przesyła wyniki do programu klienta.



Rys. 1-2 Współpraca klienta i serwera protokołu telnet

Program **`telnet`** pracuje w trybie tekstowym - obsługiwane są specjalne sekwencje znaków, które umożliwiają pewien poziom formatowania tekstu i pozycjonowania kursora w oknie klienta telnet. Serwer i klient telnetu obsługują emulację różnych typów terminali.

Najpopularniejsze to ANSI, VT-100, VT-52. Protokół telnet domyślnie używa portu 23.

Istnieje wiele programów będących klientem telnetu np. PuTTY oraz podstawowy program telnet zainstalowany w większości systemów.

Uruchomienie programu telnet:

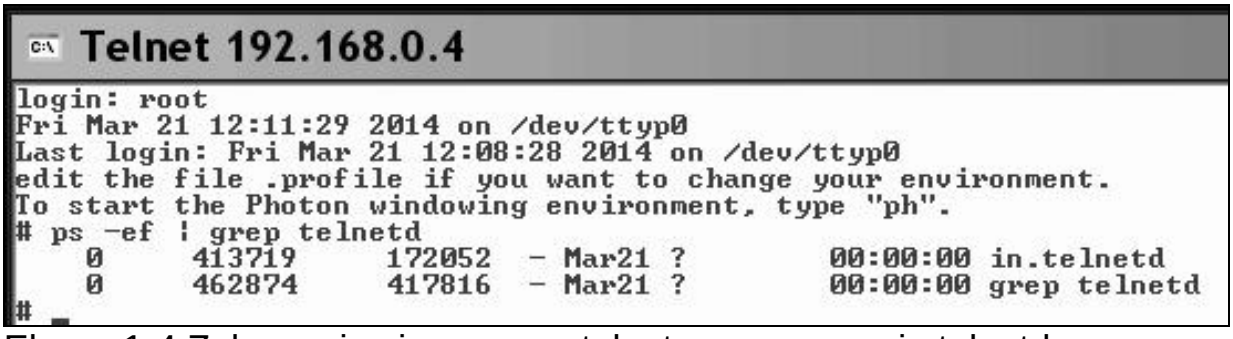
$telnet [adres_IP [nr_portu]]

Serwerem telnetu jest zwykle program **in.telnetd**. Uruchamiany jest on przez superserwer sieciowy inetd lub xinetd.

```
service telnet
{
    socket_type  = stream
    wait         = no
    user         = root
    server       = /usr/etc/in.telnetd
    bind         = 127.0.0.1
    log_on_failure += USERID
}
```

Przykład 1-6 Plik **/etc/xinetd.d/telnet** definiujący usługę telnet

Ekran 1-3 Uruchomienie programu telnet i próba połączenia z serwerem o adresie 192.168.0.4



Ekran 1-4 Zgłoszenie się serwera telnetu - programu in.telnetd

Protokół telnet posiada istotne luki bezpieczeństwa:

- Hasło przesyłane jest w sposób jawny
- Nie ma gwarancji że komunikacja zachodzi z właściwą stacją a nie z podstawioną

W związku z tym używanie protokołu poza lokalną siecią nie jest zalecane. Obecnie protokół telnet został zastąpiony przez SSH.

1.5 Usługa i protokół SSH

Po nazwą SSH (ang. *Secure Shell*) kryje się usługa będąca następcą telnetu a zapewniająca bezpieczne połączenie terminalowe ze zdalnym komputerem. Usługa ssh zastępuje starsze usługi rlogin i rsh.

W szerszym znaczeniu SSH to też nazwa rodziny szyfrowanych protokołów służących także do przesyłania plików, protokoły te to SCP (ang. *Secure Copy*) i SFTP (ang. *Secure File Transfer Protocol*). Definicja protokołu SSH powstała w grupie roboczej IETF (ang. *Internet Enginnering Task Force*) i opisano je w dokumentach RFC 4250, RFC 4251, RFC 4252, RFC 4253, RFC 4254. Protokoły SSH korzystają zazwyczaj z portu 22 połączenia TCP/IP. Transmisje SSH są szyfrowane według standardu AES choć stosowane są też standardy DES i Blowfish.

Aby wykorzystać usługę ssh należy mieć zainstalowany program klienta i serwera tej usługi. Istnieje wiele programów klienta, bardziej znane to ssh i PuTTY.

Zwykle demonem usługi jest /usr/sbin/sshd. Startowany jest przez skrypt /etc/init.d/ssh. Program może być uruchomiony z opcjami, posiada też plik konfiguracyjny: **`/etc/ssh/sshd_config`**

Po starcie nasłuchuje na porcie 22. Gdy przychodzi połączenie, za pomocą funkcji fork tworzony jest proces potomny który obsługuje nowe połączenie. Demon sshd wykonuje następujące funkcje:

- Wymiana kluczy szyfrujących
- Szyfrowanie
- Autoryzacja
- Wykonanie zdalnych poleceń
- Kopiowanie plików

```
#chkconfig -l
ssh     0:off  1:off  2:on   3:on   4:on   5:on   6:off
xinetd  0:off  1:off  2:on   3:on   4:on   5:on   6:off
xinetd based services:
        chargen:            off
        daytime:            off
        discard:            off
        echo:               off
        time:               off
```

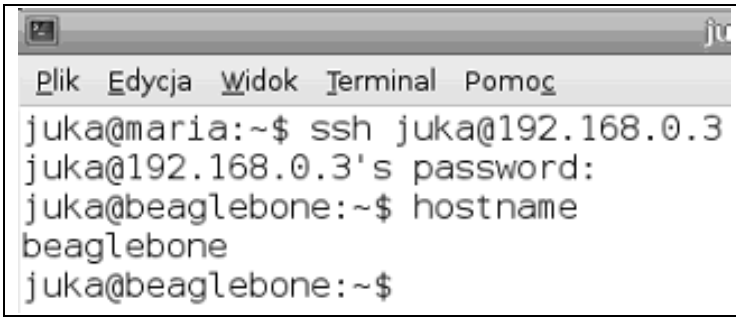
Przykład 1-7 Sprawdzenie czy usługa ssh zainstalowana

```
$ps –ef | grep sshd
root      1709     1  0 May29 ?        00:00:00 /usr/sbin/sshd
```

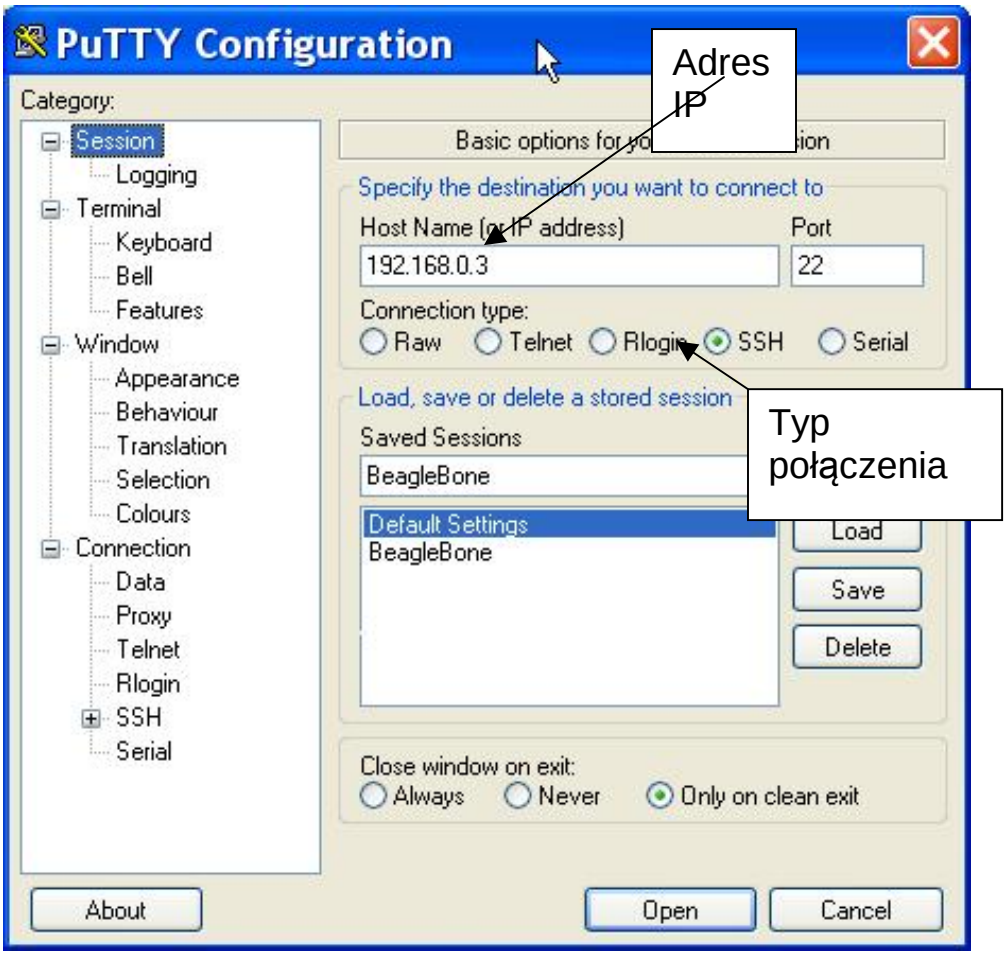
Przykład 1-8 Sprawdzenie czy demon sshd jest wykonywany

W systemie Linux możemy też użyć tekstowego klienta SSH który zazwyczaj jest już zainstalowany. Program uruchamia się:

$ssh nazwa_użytkownika@adres_ip_komputera



Ekran 1-5 Logowanie się do zdalnego komputera za pomocą klienta ssh



Ekran 1-6 Konfiguracja połączenia SSH do komputera BeagleBone Black

1.6 Przesyłanie plików, protokół FTP,

Protokół FTP jest standardowym protokołem przesyłania danych w sieci wykorzystującej TCP/IP. Umożliwia on przesyłanie plików pomiędzy komputerami z których jeden jest serwerem a pozostałe komputery, zwane klientami z tej usługi korzystają.

Pierwsza specyfikacja protokołu pojawiła się w RFC114 w roku 1971 a potem była rozwijana i dostosowana do protokołu TCP/IP co opisano w RFC765 i RFC959. Standard FTP określa dokładnie w jaki sposób oprogramowanie na jednym komputerze współdziała z programowaniem na drugim.

Interfejs użytkownika nie jest zdefiniowany, z tego też powodu może mieć różne implementacje.

Popularny interfejs standardu BSD obejmujący ponad 50 poleceń, jednak w praktyce wykorzystuje się tylko kilka. Standard FTP przewiduje obecność dwóch połączeń:

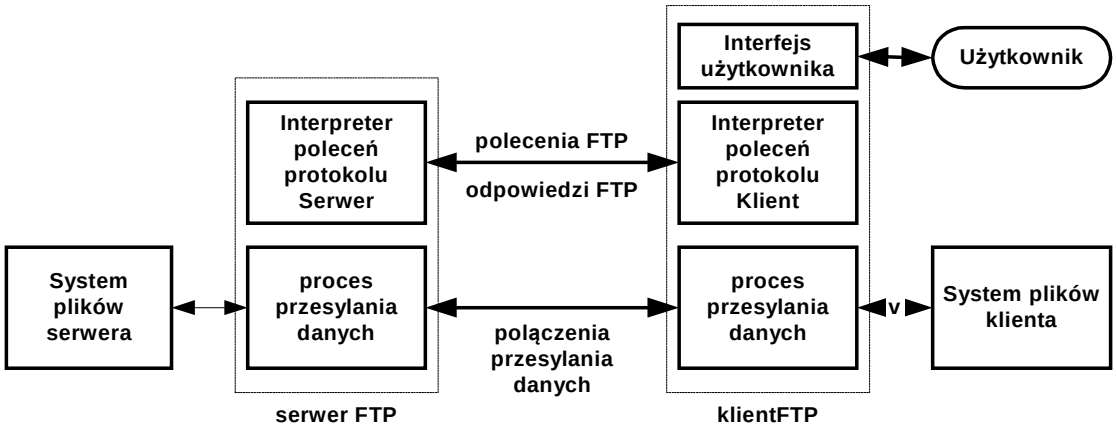
- połączenia sterującego (port 21)
- połączenia przesyłania danych (port 20).

Połączenie sterujące utrzymywane jest przez cały czas sesji, połączenie dla danych może być stosownie do potrzeb tworzone i zamykane. Dzięki istnieniu dwóch oddzielnych kanałów polecenia i przesyłane dane mogę być separowane.

Początkiem pracy jest proces autoryzacji. Polega on na wysłaniu przez klienta do serwera polecenia **open**. W odpowiedzi serwer przesyła żądanie podania nazwy użytkownika i hasła. Klient przesyła nazwę użytkownika i hasło które podlega weryfikacji przez serwer. Gdy przebiegnie ona pomyślnie, klient FTP ma takie prawa dostępu do plików serwera jakie miałby lokalny użytkownik.

Polecenia:

- Pobieranie plików z serwera na komputer klienta: **`get, mget`**
- Przesyłanie z klienta do serwera: **`put`**,**`mput, send`**.
- Listowanie zdalnego katalogu: dir
- Zamknięcie połączenia: close



Rys. 1-3 Schemat poglądowy aplikacji FTP – strona serwera i klienta

FTP umożliwia przesyłanie plików pomiędzy komputerami różnych typów o odmiennej reprezentacji danych. Możliwe są dwa podstawowe tryby przesyłania:

- tekstowy
- binarny.

W trybie tekstowym przesyłane są teksty składające się ze znaków ASCII lub EBCDIC podzielone na wiersze (możliwa translacja).

W trybie binarnym przesyłane są bajty bez możliwości translacji.

FTP może pracować w trybie aktywnym i pasywnym.

- W trybie aktywnym klient tworzy połączenie TCP do serwera.
- Tryb pasywny stosowany jest gdy klient jest poza zaporą (ang. firewall) i nie może przyjąć połączenia nawiązywanego od strony serwera.

Posługiwanie się ftp wymaga programu klienta i serwera.
Demon: /usr/sbin/in.ftpd

/usr/sbin/in.ftpd [-A | -a] [-C] [-c] [-d] [-E] [-l] [-v] [-T *maxtimeout*] [-t *timeout*] [-p *port*] [-U *ftpusers-file*] [-u *umask*] [-r *realm-file*] [-s *srvtab*] [-w{ip|*maxhostlen*[,{striplocal|nostriplocal}]}]

Serwer **ftpd** uruchamiany jest przez demona **xinetd**

```
service ftp
 {
    socket_type          = stream
    wait                 = no
    nice                 = 10
    user                 = root
    server               = /usr/sbin/in.ftpd
    server_args          = -l
    instances            = 4
    log_on_success       += DURATION HOST USERID
}
```

Przykład 1-9 Plik **/etc/xinetd.d/ftp** uruchamiania usługi ftp

Zasady autoryzacji:

1. Nazwa użytkownika musi być w pliku /etc/passw i posiadać hasło nie będące NULL
2. Nazwa użytkownika nie może pojawić się w pliku /etc/ftpusers
3. Użytkownika musi posiadać standardowy shell.
4. Jeżeli nazwa użytkownika występuje w pliku /etc/ftpchroot to system operacyjny zmienia katalog główny (chroot) na katalog domowy tego użytkownika.
5. Jeżeli nazwa użytkownika jest anonymous lub ftp to taki użytkownik musi być zarejestrowany w pliku /etc/passwd. W tym przypadku system zmienia katalog główny (chroot) na katalog domowy użytkownika ftp.

Klienci FTP

Istnieje wiele klientów ftp. Mogą pracować w trybie tekstowym, semigraficznym lub graficznym. W większości systemów (Linux, Windows) zainstalowany jest systemowy klient FTO o nazwie ftp.

```
juka@maria:~/prog/lab/sieci$ ftp 127.0.0.1
Connected to 127.0.0.1.
220 maria.localdomain FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17)
Name (127.0.0.1:juka): juka
331 Password required for juka.
Password:
```

Ekran 1-7 Wywołanie i zgłoszenie programu ftp

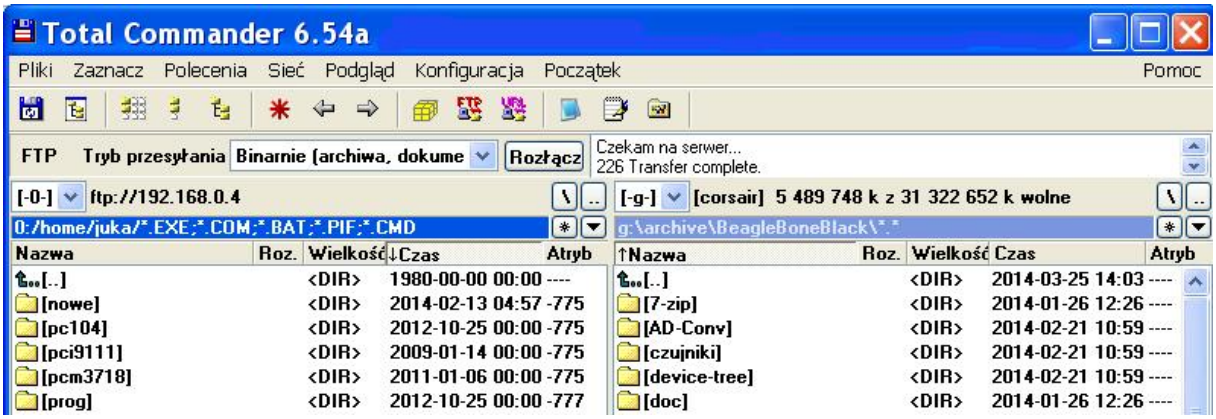
?, help	Pomoc
open	Otwarcie ołączenia
bin	Tryb przesyłania binarny
ascii	Tryb przesyłania tekstowy
cd	Zmiana katalogu na maszynie zdalnej
lcd	Zmiana katalogu na maszynie lokalnej
get plik	Pobranie pliku zdalnego i zapis w lokalnym katalogu
put plik	Wysłanie pliku i zapis w zdalnym katalogu
mget *	Pobranie wielu plików zdalnych i zapis w lokalnym katalogu
mput	Wysłanie wielu plików lokalnych i zapis w zdalnym katalogu
pwd, ls	Wypisz katalog bieżący na zdalnej maszynie
close	Zamknij połączenie
quit, by	Zakończenie sesji

Tabela 1-7 Ważniejsze polecenia programu ftp

Znacznie wygodniejsze w użyciu są programy przesyłania plików działające w trybie semigraficznym jak Midnight Commander, czy graficznym jak Total Commander czy Filezilla. Tworząc połączenie FTP należy podać adres IP serwera FTP i nazwę użytkownika. Pokazuje to przykład z Ekran 1-8 gdzie w programie Total Commander definiujemy połączenie do systemu QNX6 Neutrino wykonywanego na maszynie wirtualnej który posiada adres IP równy 192.168.0.4 a użytkownik ma identyfikator **`juka`**.

Ekran 1-8 Definiowanie połączenia FTP do systemu QNX6.3.2 na maszynie wirtualnej

Przykład kopiowania plików z komputera zdalnego na lokalny pokazuje Ekran 1-9.



Ekran 1-9 Kopiowanie plików pomiędzy systemami Windows i QNX6 Neutrino za pomocą programu Total Commander

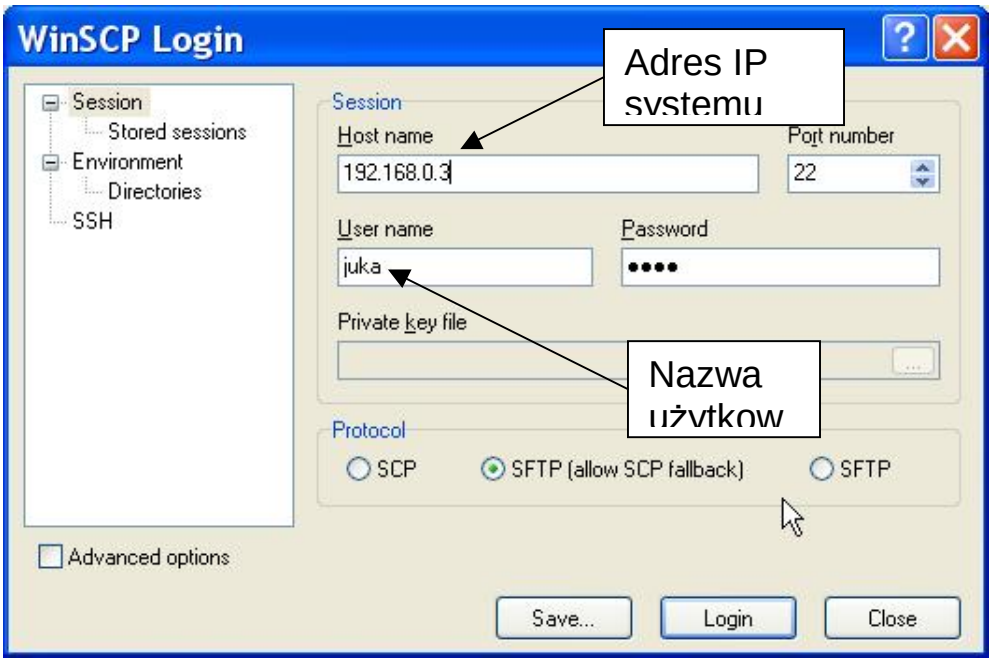
1.7 Protokół SFTP

Protokół SFTP (ang. *Secure File Transfer Protocol*) jest sieciowym protokołem służącym do zarządzania plikami iich przesyłania. Został opracowany jako rozszerzenie protokołu SSH w wersji 2.0. Zakłada on że transmisja przebiega przez bezpieczny kanał, w tym przypadku dostarczany przez SSH.

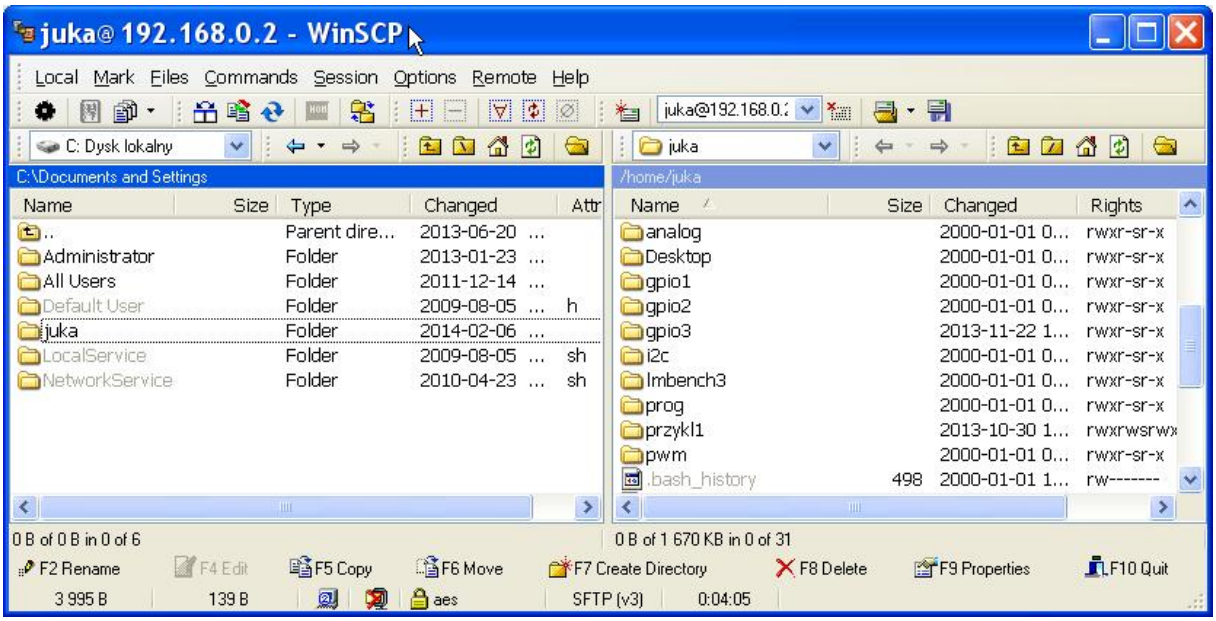
Protokół umożliwia:

- Wznawianie przerwanego transferu,
- Listowanie zawartości zdalnych katalogów,
- Usuwanie zdalnych plików
- Kopiowanie plików zachowuje atrybuty plików jak np. czas ostatniego dostępu.

Aplikacja SFTP składa się z programu klienta i serwera. Protokół jako że jest częścią SSH używa portu 22. Programy klienta mogą być w wersji konsolowej jak i korzystać z interfejsu graficznego. Bardziej znane implementacje klienta to WinSCP pracujący w systemie Windows i OpenSSH pracujący w Linuksie. Szeroko rozpowszechnioną wersją serwera SFTP jest OpenSSH pracujący w systemie Linux.



Ekran 1-10 Konfigurowanie połączenia z systemem wbudowanym BBB w programie WinSCP



Ekran 1-11 Widok programu WinSCP połączonego z systemem wbudowanym

1.8 Protokół SCP

Podstawową wadą protokołu FTP jest brak możliwości szyfrowania nazwy użytkownika, hasła a także danych. Naraża to użytkownika na podsłuchanie hasła a co zatem idzie włamania do systemu.

Wady tej pozbawiony jest protokół SCP (ang. *Secure Copy Protocol*). Jest to sieciowy protokół, oparty na wcześniejszym rozwiązaniu BSD RCP (ang. *Remote Copy Protocol*) umożliwiający przesyłanie plików między systemami. Protokół SCP do przesyłania danych, szyfrowania i uwierzytelniania wykorzystuje wspomniany wcześniej protokół SSH.

Podstawowa składnia polecenie **scp** występującego w systemie Linux w trybie konsolowym jest następująca:

```
scp [opcje][[użytkownik@]]host1:]plik1
[[użytkownik@]]host2:]plik2
```

Przykład

```
$scp bhello.c juka@192.168.0.3:.
```

```
juka@maria:~/prog/beagle$ scp bhello.c juka@192.168.0.3:.
The authenticity of host '192.168.0.3 (192.168.0.3)' can't be established.
RSA key fingerprint is 7f:25:1c:6e:fc:1d:93:d8:d6:38:a1:de:f9:7a:24:0a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.3' (RSA) to the list of known hosts.
juka@192.168.0.3's password:
bhello.c                                                      100%  135
```

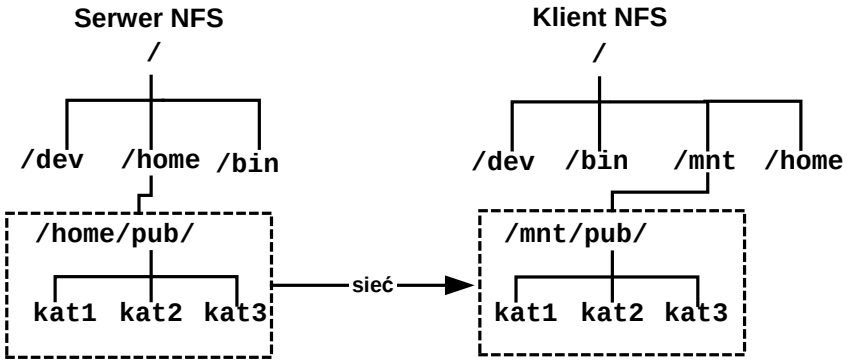
Przykład 1-10 Kopiowanie pliku bhello.c z komputera macierzystego pracującego w systemie Linux do systemu wbudowanego BBB za pomocą programu scp

2 Sieciowe systemy plików

2.1 Informacje wstępne

Sieciowe systemy plików umożliwiają udostępnienie, poprzez sieć, części systemu plików komputera innym komputerom.

- Serwer – strona która udostępnia pliki
- Klient - strona która korzysta z plików



Rys. 2-1 Zawartość katalogu /home/pub z serwera udostępniana poprzez NFS w katalogu /mnt/pub komputera klienta

Bardziej znane implementacje :

- NFS (ang. *Network File System*) – opracowany w firmie Sun Microsystems
- Samba (ang. *Server Message Block*) - umożliwia dostę do systemu plików z systemu Windows
- Andrew File System (AFS) - rozproszony system plików opracowany w Carnegie Mellon University a potem rozwijany przez IBM. Obecnie Open Source. Zawiera dodatkowe zabezpieczenia w porównaniu do NFS.
- Coda (ang. *Constant Data Availability*) – rozproszony system plików opracowany na podstawie AFS-2 w Carnegie Mellon University.

Pożądane własności:

1. Wydajność
2. Replikacja serwera
3. Bezpieczeństwo: autentykacja, szyfrowanie transmisji, kontrola dostępu
4. Odporność – działanie w warunkach częściowej awarii sieci
5. Dobrze zdefiniowana semantyka współdzielenia plików nawet w przypadku błędów sieciowych
6. Skalowalność
7. Dostępność kodu źródłowego, liberalna licencja

Omawiany będzie NFS – ze względu na popularność

NFS zrealizowany jest w Sun Microsystems, protokół otwarty. Implementacja w oparciu o RPC (ang. *Remote Procedures Calls*), zdalne wywoływanie procedur.

Wersje:

- NFSv1 – wersja eksperymentalna
- NFSv2 – zrealizowana w 1989, opisana w RFC 1094, wykorzystywany protokół UDP. Pozwala na dostęp do plików wielkości do 2 GB (ograniczenie 32 bit).
- NFSv3 – zrealizowana w 1995, opisana w RFC 1813, Pozwala na dostęp do plików wielkości ponad 2 GB (system 64 bit). Szereg ulepszeń, jako warstwa transportu wykorzystywane TCP
- NFSv4 – zrealizowana w 2000, opisana w RFC 3530. Zawiera ulepszenia wydajności i większy stopień bezpieczeństwa.
- NFSv4.1 - zrealizowana w 2010, opisana w RFC 5661. Przystosowana do serwerów klastrowych. Równoległy dostęp do plików rozmieszczonych na wielu serwerach

Platformy:
Opracowana dla systemu Unix, wykorzystywany w Linux, AIX, Solaris, FreeBSD, HP-UX. Istnieje także wersja dla Windows, Mac OS, AS-400.

Sposoby implementacji:
W pewnych implementacjach serwer NFS implementowany jest jako proces (demon) np. nfsd, w innych jako usługa jądra.

2.2 Działanie NFS

Typowa implementacja:

1. Na serwerze wykonuje się demon NFS, zwykle jest to proces **nfsd**. Udostępnia on pliki klientom.
2. Administrator systemu określa jakie foldery mają być eksportowane i na jakich warunkach. Okreslane jest to w pliku konfiguracyjnym /etc/exports i poprzez polecenie **exportfs**.
3. Administrator bezpieczeństwa systemu sprawdza czy jest w stanie rozpoznać i zweryfikować klientów.
4. Komputer klienta żąda dostępu do eksportowanych przez serwer folderów, typowo porzez polecenie mount. Klient kieruje zapytanie do serwera (**rpcbind**) których portów NFS używa serwer. Klient łączy się z serwerem **nfsd** i montuje eksportowany system plików (proces **mountd**).
5. Gdy połączenie i weryfikacja przebiegnie pomyślnie, klient uzyskuje dostęp do eksportowanych przez serwer folderów w dozwolonym zakresie.

2.3 Instalacja i konfiguracja

Instalacja

Instalacja serwera NFS
```
# apt-get install nfs-common portmap
```

Instalacja serwera NFS Linux
```
# apt-get install nfs-kernel-server nfs-common portmap
```

Konfiguracja

Konfiguracja serwera:

Są trzy pliki konfiguracyjne które trzeba edytować:

/etc/exports
/etc/hosts.allow

/etc/hosts.deny

Plik **/etc/exports:**
Plik zawiera listę pozycji udostępniania. Każda z pozycji pokazuje folder który ma być udostępniony i wskazuje sposób w jaki ma się odbywać to udostępnienie.

Ogólna postać linii definicyjnej:

Udostępniany_katalog_serwera definicja_klienta(opcja$_1$,opcja$_2$,...,opcja$_n$)

Klient może być zdefiniowany przez adres IP lub nazwę. W nazwie klienta dozwolone są znaki * . Po nazwie w nawiasie specyfikuje się opcje dostępu. Ważne aby przed nawiasem otwierającym nie wystąpiła spacja która traktowana jest jako separator klienta.

Najważniejsze opcje:

- rw – odczyt, zapis
- ro – tylko odczyt

Przykłady pliku /etc/exports:

```
/home/client1 192.168.0.101(rw,sync) www.slimak.com(ro)
/var/www      192.168.0.101(ro)
```

Gdy serwer nfs się wykonuje należy wykonać polecenie:
 # exportfs -a

Konfiguracja klienta:
Typowym postępowaniem jest użycie polecenia mount:

mount –t typ urządzenie katalog_montowania

Np:

#mount files.slimak.com:/home /mnt/nfs

Polecenie montuje z serwera **files.slimak.com** folder /home w katalogu /mnt/nfs klienta. Aby się to udało katalog /mnt/nfs po stronie klienta musi istnieć a folder /home musi być eksportowany przez serwer NFS.

Zwykle montowanie woluminów NFS odbywa się automatycznie przy starcie systemu. Montowane woluminy powinny być wyspecyfikowane w pliku /etc/fstab

Pola linii konfiguracyjnej:

- Pole 1 – specjalne urządzenie blokowe lub zdalny system plików
- Pole 2 – miejsce gdzie ma być zamontowany system plików
- Pole 3 – typ systemu plików (nfs, ext2, ext3, iso9660, msdos,...)
- Pole 4 – opcje montowania. Lista opcji oddzielona przecinkami.
- Pole 5 – uzywane przez polecenie dump
- Pole 6 – określa kolejność sprawdzania systemu plików (0 – nie sprawdzane).

Przykład pliku /etc/fstab

```
192.168.0.10:/home /home nfs
rw,rsize=4096,wsize=4096,hard,intr,async,nodev,nosuid 0 0
```

2.4 Przykład

Serwer NFS Linux Debian Squeeze, klient komputer BeagleBone Black

Instalacja serwera NFS na komputerze PC z systemem Debian Squeeze o adresie 192.168.0.4

```
# apt-get install nfs-common portmap
```

Edycja pliku /etc/exports:
Trzeba przejść do trybu root:

$su root

Uruchomić edytor tekstu np. gedit

#gedit &

Następnie zmodyfikować plik: /etc/exports jak poniżej:

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
/home/juka/  192.168.0.*(rw,sync,no_root_squash,no_subtree_check)
```

Przykład 2-1 Zawartość pliku /etc/exports

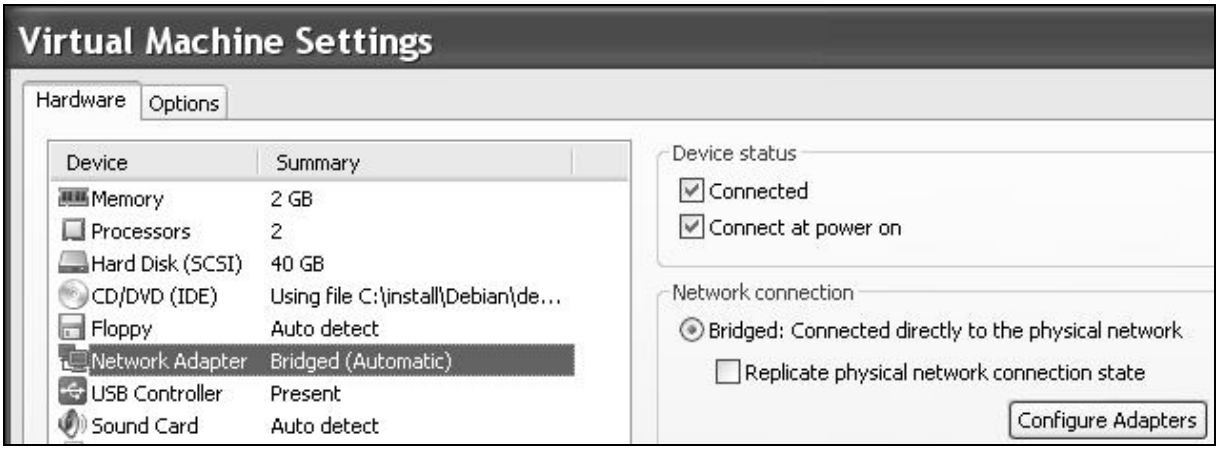
Restart serwera NFS:

```
#service portmap restart
#/etc/init.d/nfs-kernel-server restart
```

```
root@maria:/home/juka/prog/lab/sieci# service portmap restart
Stopping portmap daemon....
Starting portmap daemon....
root@maria:/home/juka/prog/lab/sieci# /etc/init.d/nfs-kernel-server restart
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon....
Exporting directories for NFS kernel daemon....
Starting NFS kernel daemon: nfsd mountd.
root@maria:/home/juka/prog/lab/sieci#
```

Ekran 2-1 Restart serwera portmap i serwera NFS

Uwaga!
Gdy serwer wykonywany jest na maszynie wirtualnej musi mieć w ustawieniach tryb: Bridged



Ekran 2-2 Ustawienia maszyny wirtualnej

Klient
Po stronie klienta – komputer BeagleBone Black z systemem Angstrom
Montujemy system plików typu nfs z komputera 192.168.0.4 folder /home do folderu lokalnego /mnt/home

#mount –onolock –t nfs 192.168.0.4:/home/juka /mnt/home

```
root@beaglebone:/mnt# mount -onolock -t nfs 192.168.0.4:/home /mnt/home
root@beaglebone:/mnt# ls /mnt/home
juka
root@beaglebone:/mnt# ls
```

Ekran 2-3 Zamontowanie sieciowego systemu plików z serwera 192.168.0.3

Prawidłowość zamontowania sprawdzamy poleceniem df:

$df

```
juka@beaglebone:~$ df
Filesystem              1K-blocks     Used Available Use% Mounted on
rootfs                    1738184  1558968     89252  95% /
/dev/root                 1738184  1558968     89252  95% /
devtmpfs                   255280        0    255280   0% /dev
tmpfs                      255408        0    255408   0% /dev/shm
tmpfs                      255408      224    255184   1% /run
tmpfs                      255408        0    255408   0% /sys/fs/cgroup
tmpfs                      255408        4    255404   1% /tmp
192.168.0.5:/home/juka   39559680 11649536  25900544  32% /mnt/home
```

Ekran 2-4 Sprawdzenie zamontowanych systemów plików

Jeśli chcemy aby sieciowy system plików był montowany po każdym restarcie należy zmodyfikować plik /etc/fstab dodając tam linię.

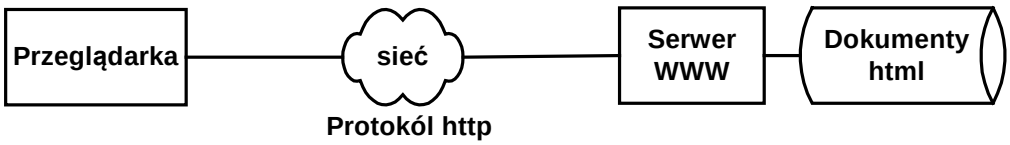
`192.168.0.4:/home/juka /mnt/home nfs defaults,nolock 0 0`

3 System WWW

3.1 Architektura WWW

WWW (ang. *World Wide Web*) jest systemem udostępniania dokumentów. Składa się z:

- Serwerów WWW udostępniających dokumenty
- Klientów w postaci przeglądarek
- Sieci pracującej w oparciu o protokół TCP/IP
- Dokumentów w postaci HTML



Rys. 3-1 Zasada działania WWW

Przeglądarka pozwala na wyświetlenie tekstu zawierającego tekst i grafikę. Niektóre informacje są wyróżnione tak aby można je było wskazać myszką. Gdy użytkownik kliknie myszką w wyróżniony element przeglądarka przekazuje nowe informacje związane ze wskazaną pozycją.

WWW jest systemem hipermedialnym. Informacja jest przechowywana jako zbiór dokumentów.

- System hipermedialny – zawiera tekst, grafikę, dźwięki, obrazy
- System hipertekstowy – zawiera tylko tekst w formacie html

Dokumenty mogą zawierać odsyłacze do innych dokumentów. System jest rozproszony tzn. wskazywane dokumenty mogą leżeć na innych komputerach administrowanych niezależnie. Stąd odsyłacze mogą być niepoprawne.

3.2 Terminologia:

HTTP (ang. *Hyper Text Transfer Protocol*)
Protokół który służy do przenoszenia informacji pomiędzy serwerem i klientem

MIME (ang. *Multimedia Interner Mail Extensions*)
Standard, pierwotnie zdefiniowany dla poczty elektronicznej, umożliwiający przesyłanie informacji innej niż tekst. Określa format danych przesyłanych przy pomocy protokołu HTTP. Standard opisany w RFC1521

HTML (ang. *Hypertext Markup Language*)
Język definiujący dokumenty udostępniane na stronach WWW. Opisuje strukturę dokumentu, układ strony, styl, obiekty osadzone w dokumencie takie jak obrazy, pliki, odnośniki.

URL (ang. *Uniform Resource Locator*).
Położenie dokumentu WWW określane jest poprzez URL jednolity adres zasobu.

3.3 Język HTML

Dokument udostępniany za pomocą WWW jest nazywany stroną WWW. Strony muszą mieć ściśle określony format aby przeglądarki mogły je interpretować. Strony zapisane są w języku HTML (ang. *Hypertext Markup Language*) czyli języku opisu struktury dokumentów hipertekstowych.

Język HTML specyfikuje:
- Zawartość dokumentu
- Strukturę – ważność tekstu, odsyłacze, itd.

Każdy dokument podzielony jest na:
- Nagłówek dokumentu
- Treść dokumentu

Każdy dokument w HTML jest plikiem tekstowym który zawiera informacje i znaczniki. Znaczniki służą do zapisanie struktury dokumentu i podają sposób jego prezentacji. Są wstawiane w miejscach w których potrzebne jest ich działanie.

Znacznik ma postać nazwy w nawiasach < NAZWA> a jego działanie kończy się jako </NAZWA>.
Każdy dokument zaczyna się znacznikiem <HTML> a kończy </HTML>.

```
<HTML>
  <HEAD>
   <TITLE>
    Tekst tytułu
   </ TITLE>
   <BODY>
   Treść dokumentu
   </BODY>
</HTML>
```

Przykład 3-1 Postać dokumentu w języku HTML

Innym znacznikiem jest
 - przejście do nowej linii.

Tytuły

Każdy dokument HTML powinien mieć tytuł. Zawarty jest on wewnątrz znaczników <TITLE> i </ TITLE>. Tytuły wyświetlane są zwykle poza tekstem, i często występują jako nazwa zakładki.

Komentarze

Komentarz ma następującą postać:
<! Treść komentarza >

Nagłówki

Język HTML definiuje 6 znaczników do wyświetlania kolejnych poziomów nagłówków. Są one postaci:

- <Hi> - początek nagłówka poziomu i
- </Hi> - koniec nagłówka poziomu i

Wyliczenia

Wyliczenie można podać w postaci listy nie uporządkowanej
<UL> - początek listy
</UL> - koniec listy
Każda pozycja ma być poprzedzona znacznikiem <LI>

```
Miasta dolnośląskie:
<UL>
<LI> Wrocław
<LI> Jelenia Góra
<LI> Legnica
<LI> Wałbrzych
</UL>
Koniec listy
```

Przykład 3-2 Lista przykładowa – postać źródłowa

Powyższy tekst zostanie wyświetlony przez przeglądarkę jako:

Miasta dolnośląskie:
- Wrocław
- Jelenia Góra
- Legnica
- Wałbrzych

Koniec listy

Przykład 3-3 Lista przykładowa – postać wynikowa

Formatowanie tekstu

Zwykle występujący w dokumentach tekst formatowany jest w sposób ciągły (nie zawiera znaków nowej linii) gdyż nie wiadomo jak szerokie pole na tekst będzie w przeglądarce. Jednakowoż pewien stopień formatowania jest dostępny.

 - znak nowej linii
<P> - Nowy akapit
<HR> - Pozioma linia rozdzielająca.

Informacja graficzna

Informacja graficzna wchodząca w skład dokumentu WWW umieszczona jest w oddzielnych plikach. Dokument zawiera odsyłacze do tych plików w postaci znaczników. Znaczniki te mają postać <IMG>.

<IMG SRC=”obrazek.gif”>

Obrazy nie zawierają informacji formatującej. Sposób wyświetlania może zawierać słowo ALIGN.

```
Obrazek. <IMG SRC=”obrazek.gif” ALIGN=midle>
```

Przykład 3-4 Postać źródłowa obrazka w języku HTML

3.4 Identyfikacja strony - adresacja URL

Identyfikacja stron WWW jest skomplikowana z następujących powodów:

- Strony mogą być zlokalizowane na różnych komputerach
- Dany komputer może przechowywać wiele stron
- Strona może być reprezentowana w różnych formatach
- Jej pobieranie może się odbywać za pomocą różnych protokołów

Położenie dokumentu WWW określane jest poprzez URL jednolity adres zasobu (ang. *Uniform Resource Locator*). Postać URL jest następująca:

```
Protokół://nazwa_komputera:port/ścieżka_lokalna/nazwa_dokumentu
```

Definicja 3-1 Format adresu URL

Numer portu jest opcjonalny, domyślnie jest to port 80.

3.5 Odsyłacze hipertekstowe pomiędzy dokumentami

Najważniejszą cechą dokumentów hipertekstowych jest możliwość wskazania innego dokumentu który może być położony na lokalnym lub innym komputerze. Odbywa się to poprzez odsyłacze hipertekstowe. Odsyłacz jest odnośnikiem pasywnym to znaczy nie powoduje on samoczynnego ściągnięcia innego dokumentu (tak jak np. znacznik IMG) ale jedynie jego wyróżnienie. Gdy użytkownik wybierze ten dokument nastąpi jego sprowadzenie.

Znacznik używany do oznaczenia odsyłaczy nazywany jest zakotwiczeniem (ang. *anchor*). Ma on postać pary <A> i </A>. Znacznik <A> zawiera adres URL wskazywanego dokumentu. Przeglądarka wyróżnia tekst oznaczony jako odsyłacz. Poniżej podany jest przykład.

```
Tekst wykładu jest dostępny  na stronie IIAR
 <A HREF=”http://www.iiar.pwr.wroc.pl”  </A>.
```

Przykład 3-5 Postać źródłowa odnośnika

Tekst wykładu jest dostępny na stronie IIAR Politechniki Wrocławskiej

Przykład 3-6 Postać wyświetlana dokumentu

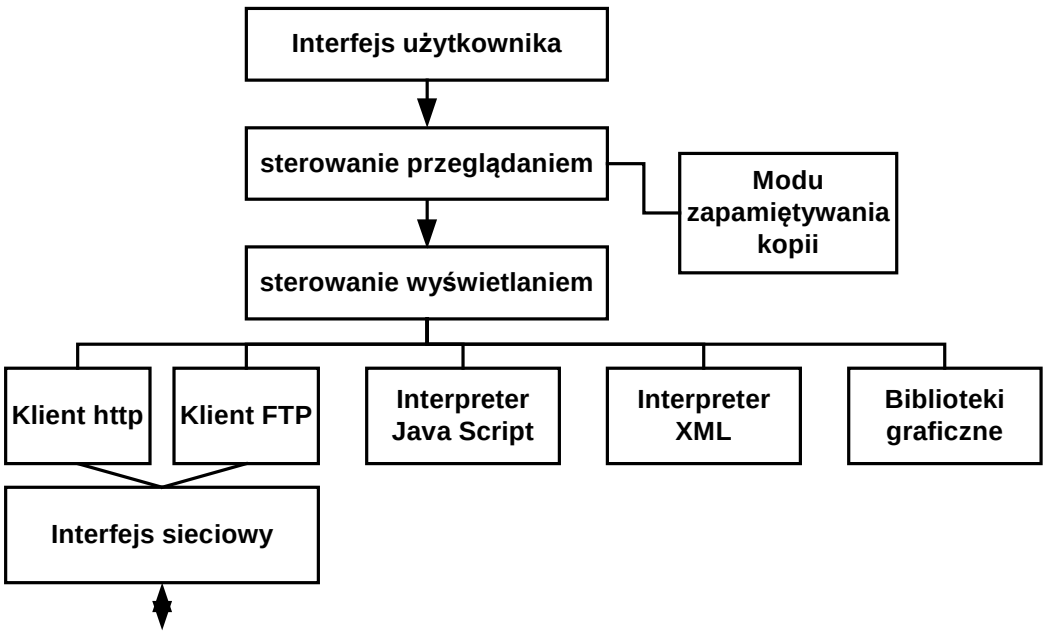
3.6 Współpraca przeglądarki i serwera WWW

Współpraca przeglądarki i serwera WWW odbywa się poprzez protokół TCP w trybie połączeniowym. Po wskazaniu przez użytkownika adresu URL dokumentu przeglądarka łączy się z serwerem (port 80), pobiera treść dokumentu i się rozłącza. Pobieranie dokumentu odbywa się przy pomocy protokołu HTTP (ang. *Hyper Text Transfer Protocol*)

3.7 Przeglądarka WWW

Przeglądarka wysyła do serwera żądania przesłania dokumentów spod danego URL a następnie interpretuje przesyłane przez serwer dane. Składa się ona z następujących modułów:

- Moduł sterujący – interpretacja danych z klawiatury i myszki oraz wywoływanie innych modułów
- Interpreter języka HTML i inne opcjonalne interpretery (np. Java Script, PDF,XML)
- Klienta HTTP i klientów innych protokołów (np. FTP)
- Moduł wyświetlający – biblioteki dostępu do funkcji graficznych.



Rys. 3-2 Struktura przeglądarki WWW

Ważną funkcją przeglądarki jest obsługa obiektów wskazywanych na dokumencie przez użytkownika. Przeglądarka musi określić związek pomiędzy pozycją kursora na ekranie a obiektem dokumentu a następnie wykonać akcję związaną ze sprowadzeniem i wyświetleniem wskazanego obiektu.

4 Protokół HTTP

4.1 Przeznaczenie

Protokół HTTP jest protokołem specyfikującym sposób udostępniania dokumentów WWW i sposób współpracy pomiędzy przeglądarką a serwerem WWW. Występuje w wersji 1.0 i 1.1.
Wymiana danych odbywa się poprzez gniazdka TCP/IP.
Transmitowane są zasoby (ang. *resources*). Zasoby zwykle odpowiadają plikom ale mogą też być generowane przez serwer dynamicznie (np. przez CGI).

4.2 Transakcja HTTP

Współpraca klienta i serwera

Współpraca pomiędzy przeglądarką a serwerem odbywa się w trybie klient - serwer. Zasadnicze kroki protokołu są następujące:

- Klient HTTP tworzy połączenie
- Klient wysyła żądanie przesłania zasobu
- Serwer HTTP przesyła żądane dane
- Serwer zamyka połączenie

Zarówno żądanie jak odpowiedź mają postać:

- Linia początkowa
- Zero lub więcej linii nagłówka
- Pustej linii
- Treści komunikatu

```
<linia poczatkowa , różna dla żądania i odpowiedzi>
Nagłówek1:wartość1
Nagłówek2:wartość2
Nagłówek3:wartość3
...
NagłówekN:wartośćN

<opcjonalna treść komunikatu; może zawierać wiele linii a
także dane binarne >
```

Linia początkowa i nagłówki powinny się kończyć znakami CR, LF.

Linia początkowa

Linia początkowa zawiera trzy składniki oddzielone spacjami:

- Nazwa metody
- Lokalna ścieżka do zasobu
- Oznaczenie wersji protokołu HTTP

Polecenie GET

Polecenie GET oznacza żądanie pobrania zasobu który wyspecyfikowany jest jako drugi parametr. Ścieżka jest oznaczeniem lokalnego zasobu (URI) – występuje w URL po nazwie komputera. Oznaczenie wersji protokołu ma postać: "HTTP**/**x.x",

```
GET /path_to_file/index.html HTTP/1.0
```

Linia początkowa odpowiedzi

Linia początkowa odpowiedzi zawiera trzy składniki oddzielone spacjami:

- Oznaczenie wersji protokołu HTTP – tak jak w żądaniu
- Kod odpowiedzi – liczba 3 cyfrowa wskazująca na przyczynę błędu
- Frazę opisu przyczyny – podaje czytelny opis przyczyny błędu

```
HTTP/1.0 200 OK
HTTP/1.0 404 Not Found
```

Przykłady linii początkowej odpowiedzi

1xx	Tylko informacja
2xx	Sukces
3xx	Skierowanie klienta do innego URL
4xx	Błąd po stronie klienta
5xx	Błąd po stronie serwera

Tab. 0-1 Kody błędów odpowiedzi protokołu HTTP

Linie nagłówka

Linie nagłówka podają informacje o żądaniu lub odpowiedzi lub też specyfikują zawartość żądania/odpowiedzi. Są one w postaci tekstowej formatu: **`Nazwa_nagłówka:wartość`** i kończą się znakami CR i LF.

Protokół HTTP 1.0 definiuje 16 nagłówków a HTTP 1.1 definiuje 46 nagłówków.
Przykład nagłówka:

```
User-agent: Mozilla/3.0Gold
```

Treść komunikatu

Po nagłówkach komunikatu występuje (opcjonalnie) treść komunikatu. Gdy komunikat zawiera treść zwykle występuje nagłówek który stanowi jej opis i podaje zawartość i długość.

Content-Type:zawartość (np. text/html lub image/gif)
Content-Length:długosc (np. 2345)

4.3 Przykład transakcji HTTP

Sprowadzamy zawartość URL: http://www.somehost.com/path/file.html

1. Otwieramy połączenie z komputerem www.somehost.com port 80.
2. Wysyłamy żądanie:

```
GET /path/file.html HTTP/1.0
From: uzytkownik@xxx.com
User-Agent: HTTPTool/1.0
[pusta linia]
```

3. Serwer powinien odpowiedzieć (przykładowo):

```
HTTP/1.0 200 OK
Date: Fri, 16 Dec 2011 22:09:34 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Witamy!</h1>
(kolejne linie)
  .
  .
  .
</body>
</html>
```

4.4 Inne metody HTTP

Metoda HEAD

Metoda HEAD jest podobna do metody GET z tą różnicą że nie podajemy treści komunikatu. Odpowiedź także nie może zawierać treści. Metoda służy do pobrania informacji o zasobie bez pobierania samego zasobu.

Metoda POST

Metoda POST jest używana do przesłania pewnych danych od klienta do serwera.

- Blok danych jest przesyłany w treści komunikatu
- Pole URL określa nie dane ale zwykle program który ma przetworzyć wysyłane dane.
- Odpowiedź serwera zawiera wyniki tego programu a nie statyczny plik

```
POST /login.jsp HTTP/1.1
    Host: www.mojadres.net
    User-Agent: Mozilla/4.0
    Content-Length: 27
    Content-Type: application/x-www-form-urlencoded

    userid=joe&password=guessme
```

Przykład działania metody POST

4.5 Eksperymenty z HTTP

Prosty eksperyment można przeprowadzić używając programu **telnet**. Łączymy się z serwerem www.jakis_komputer.com z portem 80.

```
telnet www.jakis_komputer.com 80
```

Następnie wprowadzamy żądanie:

```
GET /path/file.html HTTP/1.0
[nagłówki o ile są]
[pusta linia]
```

Następnie obserwujemy odpowiedź.

4.6 Kodowanie treści komunikatu

Gdy w treści komunikatu występują dane binarne powinny one być zakodowane aby przypadkowo nie wprowadzić w błąd serwera lub przeglądarki. W HTTP stosuje się kodowanie URL (ang. *URL encoding*). Opisano to w RFC 2396.

Główne zasady kodowania URL:

1. Niebezpieczne znaki koduje się jako %xx gdzie xx jest kodem HEX znaku. Niebezpieczne znaki to **=**, **&**, **%**, **+**, oraz znaki niedrukowalne.
2. Spacje zamienia się na znak +.
3. Łańcuchy nazwa i wartość łączy się za pomocą znaków = i &. Na przykład: name1=value1&name2=value2.

5 Literatura

[1] Aoki Osamu Debian Reference, https://www.debian.org/doc/manuals/debian-reference/index.en.html

[2] Douglas E. Commer, Sieci komputerowe i intersieci, WNT Warszawa 1999.

[3] HTTP Made Really Easy A Practical Guide to Writing Clients and Servers, http://www.jmarshall.com/easy/

[4] Alan Grosskurth, Michael W. Godfrey, Architecture and evolution of the modern web browser, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

[5] Prosty serwer WWW http://www.paulgriffiths.net/program/c/webserv.php

[6] Sven Goldt, Sven van der Meer, Scott Burkett , Matt Welsh , The Linux Programmer's Guide, http://www.linuxpl.org/LPG/lpg.html

[7] RFC 318 ftp://ftp.rfc-editor.org/in-notes/rfc318.txt

[8] Manual systemu Linux, http://www.kernel.org/doc/man-pages

[9] Sourceforge http://nfs.sourceforge.net.

[10] Debian Help http://www.debianhelp.co.uk/nfs.htm

[11] Wikipedia http://en.wikipedia.org/wiki/Network_File_System