1. Ustanawianie ograniczeń na użycie zasobów

W każdym systemie komputerowym zasoby potrzebne do tworzenia i wykonywania procesów są ograniczone.

W przypadku gdy w systemie działa wiele procesów ważną rzeczą jest zabezpieczenie systemu przed wyczerpaniem zasobów spowodowanym przez nadmierne zużycie zasobów przez procesy wchodzące w skład aplikacji.

☞ W bezpiecznym systemie operacyjnym powinien istnieć mechanizm limitujący pobieranie zasobów przez procesy.

System Linux posiada mechanizmy pozwalające na ustanowienie limitu na takie zasoby jak:

- czas procesora,
- pamięć operacyjna,
- pamięć wirtualna
- wielkość pamięci pobranej ze sterty,
- wielkość segmentu stosu,
- maksymalna liczba deskryptorów plików,
- maksymalna wielkość pliku utworzonego przez proces
- maksymalna liczba procesów potomnych tworzonych przez proces.
- Maksymalna liczba blokad plików i obszarów pamięci operacyjnej

Dla każdego z tych zasobów istnieje:
- ograniczenie miękkie (*ang. soft limit*)
- ograniczenie twarde (*ang. hard limit*).

Ograniczenie miękkie może być zmieniane przez proces bieżący ale nie może przekroczyć twardego.

Ograniczenie twarde może być zmieniane przez proces o statusie administratora.

1.1 Testowanie i ustawianie limitu zasobów z poziomu programu

Do testowania limitów zasobów służy funkcja **`getrlimit`**.

`getrlimit` – pobranie aktualnego limitu zasobów

```
int getrlimit(int resource, struct rlimit *rlp)
```

Gdzie:

`resource`	Określenie zasobu.
`rlp`	Wskaźnik na strukturę zawierającą bieżące i maksymalne ograniczenie.

Funkcja zwraca **`0`** gdy sukces a **`–1`** gdy błąd.

Jako pierwszy parametr funkcji podać należy numer testowanego zasobu które podaje tabela. Funkcja powoduje skopiowanie do struktury **`rlp`** aktualnych ograniczeń.

Struktura ta zawiera co najmniej dwa elementy:
`rlim_cur` - zawiera ograniczenie miękkie
`rlim_max` zawierający ograniczenie twarde.

Do ustawiania limitów zasobów służy funkcja **`setrlimit`**.

`setrlimit` – ustanowienie nowego limitu zasobów

```
int setrlimit(int resource, struct rlimit *rlp)
```

Funkcja zwraca **`0`** gdy sukces a **`–1`** gdy błąd.

Gdy proces próbuje pobrać zasoby ponad przydzielony limit system operacyjny może:

1. Zakończyć proces.
2. Wysłać do niego sygnał .
3. Zakończyć błędem funkcję pobierającą dany zasób.

Oznaczenie	Opis	Akcja przy przekroczeniu
RLIMIT_AS	Pamięć wirtualna	Wysłanie sygnału SIGSEGV do procesu przekraczającego zasób
RLIMIT_CORE	Pamięć operacyjna	Zakończenie procesu z zapisaniem na dysku obrazu pamięci operacyjnej.
RLIMIT_CPU	Czas procesora	Wysłanie sygnału SIGXCPU do procesu przekraczającego zasób.
RLIMIT_DATA	Wielkość pamięci pobranej ze sterty.	Funkcja pobierająca pamięć kończy się błędem.
RLIMIT_FSIZE	Maksymalna wielkość pliku utworzonego przez proces. Gdy 0 to zakaz tworzenia plików.	Wysłanie sygnału SIGXFSZ do procesu przekraczającego zasób. Gdy sygnał jest ignorowany to plik nie zostanie powiększony ponad limit.
RLIMIT_NOFILE	Maksymalna liczba deskryptorów plików tworzonych przez proces.	Funkcja tworząca ponad limitowe pliki skończy się błędem.
RLIMIT_STACK	Maksymalny rozmiar stosu	Wysłanie sygnału SIGSEGV do procesu przekraczającego stos.
RLIMIT_NPROC	Maksymalna liczba procesów potomnych tworzonych przez proces.	Procesy przekraczające limit nie będą utworzone.
RLIMIT_MSGQUEUE	Pamięć zajmowana przez kolejki komunika tów POSIX	

Tab. 1-1 Zestawienie niektórych zasobów systemowych podlegających ograniczeniu

Ustanowienie ograniczenie **RLIMIT_CPU** na czas zużycia procesora w systemach działających nieprzerwanie nie ma dużego zastosowania. Powodem jest fakt że jeżeli proces ma działać w nieskończoność to limit ten musi być znaczny. Tak więc system operacyjny zareaguje dopiero wtedy gdy ten limit zostanie przekroczony a w tym czasie inne procesy mogły nie uzyskać potrzebnego im czasu procesora.

Odpowiednim rozwiązaniem tego problemu jest szeregowanie sporadyczne które narzuca limit na zużycie czasu procesora w przesuwającym się do przodu oknie czasowym.

```
#include <stdlib.h>
#include <sys/resource.h>
int main(int argc, char *argv[]) {
  int res, i, num = 0;
  struct rlimit rl;
  printf("       CUR          MAX \n");
  getrlimit(RLIMIT_CPU,&rl);
  printf("CPU    %d   %d \n",rl.rlim_cur, rl.rlim_max);
  getrlimit(RLIMIT_CORE,&rl);
  printf("CORE   %d   %d \n",rl.rlim_cur, rl.rlim_max);
  rl.rlim_cur = 2;
  setrlimit(RLIMIT_CPU,&rl);
  while (1);
  return 0;
}
```

Program 1-1 Program **rlimit.c** testujący i nakładający ograniczenia na pobierane przez proces zasoby

Gdy przydzielony czas procesora ulegnie wyczerpaniu proces zakończy się z komunikatem:

$CPU time limit exceeded (core dumped)

1.2 Testowanie i ustawianie limitu zasobów z poziomu shell

Do testowania i ustawiania poziomu zużycia zasobu przez uzytkownika służy polecenie ulimit

`ulimit [-acdfHlmnpsStuv] [`*`limit`*`]`

Opcje:

-S	Zmień lub pokaż miękkie ograniczenie
-H	Zmień lub pokaż twarde ograniczenie
-a	Pokaż wszystkie ograniczenia
-c	Maksymalna wielkość pamięci operacyjnej
-d	Maksymalna wielkość segmentu danych
-f	Maksymalna wielkość tworzonego pliku
-l	Maksymalna wielkość pamięci operacyjnej która może być zablokowana
-n	Maksymalna liczba deskryptorów plików
-p	Maksymalna wielkość bufora na łącza nienazwane
-s	Maksymalna wielkość stosu
-t	Maksymalna wielkość jednostek czasu procesora
-u	Maksymalna liczba tworzonych przez użytkownika procesów
-v	Maksymalna wielkość pamięci wirtualnej dla procesu

```
$ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 16382
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) unlimited
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
$ulimit –Sn 10
$ulimit –n
10
```

Przykład 1-1 Testowanie i ustawianie limitów zasobów

Ustawiane limity zmieniane są w jednostkach 1024 bajtowych z wyjątkiem:
–t – sekundy,
-p – bloki 512 bajtów
-n – sztuki
-u - sztuki

1.3 Ustawianie limitu zasobów przy starcie systemu operacyjnego

Do ustawiania poziomu zużycia zasobu przez użytkowników służy plik:

`/etc/security/limits.conf`

```
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - an user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#        - NOTE: group and wildcard limits are not applied to root.
#          To apply a limit to the root user, <domain> must be
#          the literal username root.
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max mem. used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to values:
#          [-20, 19]
#        - rtprio - max realtime priority
#        - chroot - change root to directory (Debian-specific)
#
#<domain>      <type>  <item>         <value>
```

Przykład 1-2 Przykład pliku limits.conf

quota - Display disk usage and limits