Spis treści

Na podstawie:

- *M.J.Rochkind - Programowanie w systemie UNIX dla zaawansowanych (Advanced UNIX Programming)*
- `https://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:so:2017:labs:lab_intro`
- *Graham Glass, King Ables - Linux dla programistów i użytkowników*

1 Pojęcia podstawowe potrzebne do zrozumienia tematu

1.1. Definicje podstawowe

Limit procesu

Kernel wymusza dla procesów pewne ograniczenia, takie jak np. liczba deskryptorów otwartych plików, maksymalny rozmiar stosu, użycie CPU, itp. Takie ograniczenia nazywane są **limitami procesu**. Wartości tych ograniczeń mogą być zmieniane w pewnym stopniu przez użytkowników. Limity mogą być wyświetlane w konsoli przy użyciu polecenia *ulimit*.

Limit obecny (soft limit lub inaczej current limit)

Jest to limit, który może zostać ustawiony przez normalnego użytkownika w normalnym procesie przez wywołania systemowe. Limit ten nie może mieć jednak wartości większej od tzw. hard limitu. Przekroczenie tego limitu nie powoduje błędu, jest to tylko swego rodzaju 'ostrzeżenie'. Normalny user może więc zmniejszać i zwiększać soft limit, z tym, że nie może ustawić soft limitu na wartość większą niż aktualnie ustawiony hard limit.

Limit maksymalny (hard limit)

Jest to największa wartość jaką może mieć ustawiony soft limit. Proces nigdy nie może przekroczyć wartości tego limitu, jego przekroczenie skutkuje błędem lub wygenerowaniem sygnału. Normalny user nie może zwiększać hard limitu, ale może go zmniejszać. Super user może zarówno zwiększać jak i zmniejszać soft i hard limity.

Przekroczenie limitu procesu

Przekroczenie limitu procesu zazwyczaj spowodowane jest wywołaniem funkcji, np. write (gdy zapisujemy zbyt wiele danych do pliku) albo malloc (gdy przekraczamy możliwą dostępną dla procesu pamięć). W wyniku przekroczenia hard limitu procesu, funkcja zwraca błąd lub generowany jest sygnał (np. SIGSEGV to sygnał generowany w przypadku przepełnienia stosu). W wyniku przekroczenia soft limitu włączany jest tzw. timer, który liczy czas wykonywania procesu powyżej soft limitu, jeśli czas ten osiągnie pewną określoną wartość (zazwyczaj defaultowo 7 dni), to wtedy przekroczony soft limit jest traktowany jak przekroczony hard limit i generowany jest błąd.

- Czym jest limit procesu ?
- Czym jest soft limit ? Kto może go modyfikować ? Jaką może mieć wartość ? Co powoduje jego przekroczenie ?
- Czym jest jest hard limit ? Kto i jak może go modyfikować ? Jaką może mieć wartość ? Co powoduje jego przekroczenie ?

1.2. Funkcje getrlimit, setrlimit

Wywołania systemowe umożliwiające ustawienie wartości ograniczeń to funkcje *getrlimit* oraz *setrlimit*.

```
#include <sys/resource.h>
int getrlimit(
  int resource, /* resource */
  struct rlimit *rlp /* returned limits */
); /* Returns 0 on success or -1 on error (sets errno) */
```

```
#include <sys/resource.h>
int setrlimit(
  int resource, /* resource */
  const struct rlimit *rlp /* limits to set */
); /* Returns 0 on success or -1 on error (sets errno) */
```

Jak widać wywołania te przyjmują jako argumenty liczbę int oznaczającą zasób, którego dotyczy ograniczenia oraz strukturę, która zawiera dwa pola - soft i hard limit.

- Co umożliwiają funkcje getrlimit i setrlimit ?
- Jakie argumenty przyjmują funkcje getrlimit i setrlimit ?

1.3. Zasoby jako liczby int

Do funkcji getrlimit i setrlimit przekazujemy liczbę int oznaczającą zasób, którego dotyczyć będzie ograniczenie. Liczby te są reprezentowane przez makra zdefiniowane w <sys/resource.h>.
Standardowo w implementacjach zdefiniowane jest następujących 7 zasobów:

- RLIMIT_CORE - maksymalny rozmiar pliku core (?)
- **RLIMIT_CPU** - maksymalny czas CPU w sekundach
- RLIMIT_DATA - maksymalny rozmiar segmentu danych
- **RLIMIT_FSIZE** - maksymalny rozmiar pliku w bajtach, przekroczenie go generuje sygnał SIGXFSZ
- **RLIMIT_NOFILE** - maksymalna liczba deskryptorów pliku, które może używać proces
- RLIMIT_STACK - maksymalny rozmiar stosu
- RLIMIT_AS - maksymalny całkowity rozmiar pamięci

- Jak oznaczamy zasoby przekazywane do funkcji setrlimit, getrlimit ?
- Co oznaczają flagi RLIMIT_CPU, RLIMIT_FSIZE, RLIMIT_NOFILE ?

1.4. Struktura rlimit

Jest to struktura przekazywana do funkcji setrlimit oraz getrlimit jako drugi argument. Zawiera ona dwa pola - są to soft limit i hard limit.

```
struct rlimit {
  rlim_t rlim_cur; /* current (soft) limit */
  rlim_t rlim_max; /* maximum (hard) limit */
};
```

1.5. argc, argv

To argumenty funkcji main, pierwszym argumentem jest zawsze nazwa programu, jest to pierwszy element tablicy argv, argc zawiera długość tablicy argv.

1.6. atoi

Funkcja atoi konwertuje string na int.

```
int atoi(const char * string)
```

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

int main()
{
    char * napis = "\n\t 2004u";
    int numer;
    numer = atoi(napis);
    printf("Liczba typu int: %d, oraz jako ciag znakow: %s\n", numer, napis);
    return 0;
}
```

Daje w wyniku:

```
Liczba typu int: 2004, oraz jako ciag znakow:
         2004u
```