

1. Zapoznaj się ze stroną manuala dotyczącą komendy **ps**. Sprawdź działanie ważniejszych opcji. Określ numer procesu, numer procesu rodzica, zajętość pamięci, czas uruchomienia oraz postać linii komend procesu odpowiedzialnego za połączenie z systemem. Zwróć uwagę na opcję **-o**. Zaproponuj taką postać komendy **ps**, aby listing zawierał informacje o właścicielu procesu, numerze procesu, wartościach priorytetu i parametru NICE oraz terminala, z którym proces jest połączony (przykład: **ps -eo "%p %y %c"**).
2. Utwórz w swoim katalogu domowym katalog **c3**. Przejdź do katalogu **~/c3**. Utwórz w nim plik o nazwie **prog.c**, zawierający następujący tekst programu w języku C:

```
int main (int c, char *d[]){
    double a, b;
    while (1) {
        a = 12.345543;
        b = 0.456456;
        a = a*b;
        b = a/b;
    }
    return 0;
}
```

Skompiluj kod źródłowy programu poleceniem **gcc -o prog prog.c**. Uruchom otrzymany program w tle. Ile pamięci zajmuje ten proces, jakie są wartości priorytetu i parametru NICE tego procesu. Obniż wartość jego parametru NICE korzystając z komendy **renice**. Porównaj wartości priorytetu i parametru NICE w obu przypadkach. Sprawdź procent wykorzystania czasu procesora przez Twoje procesy, wartości priorytetu i parametru NICE w kilku odstępach czasu, gdy pracują one z „normalnym” i obniżonym priorytetem. Jakie zależności występują między wartością parametru NICE, a priorytetem. Wylistuj wszystkie programy pracujące pod kontrolą bieżącej powłoki (**jobs**). Usuń uruchomione programy (**kill %....**).

3. Uruchom program **prog** ponownie, tym razem z obniżoną wartością parametru NICE (komenda **nice**) i w tle. Jakie wartości priorytetu i parametru NICE posiada zadanie teraz. Zakończ wykonywanie uruchomionego programu.

4. Uruchom program **prog** jako pierwszoplanowy. Przenieś go do tła. Sprawdź, czy program się wykonuje. Odłącz się od systemu i podłącz ponownie. Czy program jeszcze się wykonuje? Dlaczego?

5. Uruchom program **prog** w tle tak, aby wykonywał się on mimo odłączenia się od systemu (komenda **nohup**). Gdzie mamy szukać tego co program wypisuje na ekran? Usuń uruchomiony proces.

6. Uruchom program **prog** w tle. Wyślij do niego sygnał **SIGSTOP**. Jaki jest stan uruchomionego procesu? Jak z czasem zmienia się stopień wykorzystania procesora przez ten proces? Do zatrzymanego procesu wyślij sygnał **SIGCONT**. Jaki jest teraz stan procesu? Co dzieje się z wartością w kolumnie opisującej stopień wykorzystania procesora? Usuń uruchomiony proces.

7. W katalogu **~/c3** utwórz plik o nazwie **signal.c**, zawierający kod źródłowy programu w języku C zamieszczony poniżej (do pobrania poleceniem: **wget galaxy.agh.edu.pl/~boryczko/signal.c** ). Skompiluj go poleceniem: **gcc -o signal signal.c**.

```
#include <signal.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

void obsluga_sygnalu(int sig){
    printf("Proces: dostalem sygnal %d\n", sig);
}

int main(int argc, char *argv[]){
    for ( i = 1; i < 64; i++ )
        signal(i, obsluga_sygnalu); /*podl. F-c obslugujacej*/
    while(1) {
        printf ("ciagle dzialam ...\n");
        sleep(1); /* oczekuj nieskonczenie wiele razy po 1 sek. */
    }
    return 0;
}
```

Skompiluj program poleceniem: **gcc -o signal signal.c**. Uruchom program. Co stanie się z procesem, jeśli wyślemy do niego sygnał **SIGQUIT**. Jaka jest reakcja programu na ten sygnał? Jak proces zareaguje na sygnał **SIGTSTP**. Przejdź do drugiej konsoli. Ustal numer uruchomionego programu. Wyślij do niego sygnał **SIGTERM**. Jak zakończyć wykonywanie procesu? Funkcje obsługi których sygnałów nie mogą zostać podmienione?

8. Do czego służy komenda **at**. Przy pomocy komendy **at** wymuś uruchomienie programu **prog** za 2 minuty od momentu podania komendy. Sprawdź różne sposoby podania czasu uruchomienia zadania.

9. Zapoznaj się ze stronami manuala dotyczącymi komendy **crontab**. Do czego służy ta komenda? Jakiej opcje? Jaka jest składnia linii „tablicy zegara”?