Systemy operacyjne

Laboratorium (Procesy i sygnały cz. 1)

Tutaj znajduje się przykład użycia funkcji fork()
Tutaj znajduje się program rodzic.c.
Tutaj znajduje się program potomek.c.

Proces jest wykonywaniem programu. Procesy mają swoje identyfikatory **PID** będące liczbami naturalnymi. Zadanie (job) jest pojedynczym procesem lub całym potokiem (ciągiem współbieżnych procesów połączonych wejściem/wyjściem standardowym) uruchomionym z linii poleceń. Zadania uruchamiane w danym terminalu tekstowym jako identyfikatory otrzymują początkowe liczby naturalne (w poleceniach są one poprzedzane znakiem %).

Zadanie 1 (0.3 pkt)**.** Przeczytaj opisy poleceń: **ps, bg, fg, jobs, kill** oraz **trap**. Ponadto zapoznaj się z działaniem kombinacji klawiszy **ctrl+c** oraz **ctrl+z**. Jak podejrzeć wszystkie uruchomione procesy? Jak zobaczyć tylko te procesy co są uruchomione w danym terminalu tekstowym? Jak zobaczyć te procesy, które nie są bezpośrednio związane z danym terminalem tekstowym (wyświetla polecenia uruchomione w oprócz bieżącym terminalu te spoza innych)? Utwórz skrypt w Bash'u, wyświetlające w pętli nieskończonej jakiś napis co kilka sekund (skorzystaj ze **sleep**). Uruchom w tle ten skrypt. Następnie zatrzymując go, wypróbuj działanie poleceń **bg, fg** oraz **kill**. Jakie są obserwacje? Na wszystkie pytania odpowiedz w pliku tekstowym.

Zadanie 2 (0.3 pkt)**.** Utwórz w języku **ANSI C** program, który będzie rozwidlał się na dwa podprogramy, w których proces potomny będzie obliczał funkcję $f(n) = 3n^2 - 2n + 4$, a proces macierzysty będzie czekał na zakończenie procesu potomnego (zastanów się jak jeden proces może czekać na drugi). Rozważ przypadek, co zrobić jeśli polecenie **fork()** zakończy się niepowodzeniem. Przykładowe wywołanie programu:

```
./a.out
PROCES RODZICIELSKI:Witaj w procesie rodzicielskim czekam na wynik ...
PROCES POTOMNY:Witaj w procesie potomnym!
PROCES POTOMNY:Podaj liczbe n: 100
PROCES POTOMNY:Wynik to: 29804
PROCES RODZICIELSKI:Poczekalem i teraz zakoncze program!
```

Zadanie 3 (0.5 pkt)**.** Utwórz program w języku **ANSI C**, który będzie uruchamiał n procesów potomnych. Każdy z nich ma wyświetlić swój identyfikator procesu **PID** oraz jego ojca **PPID**. Przykładowe wywołania programu:

```
./a.out
PROCES RODZICIELSKI: 12406
Podaj n: 3
Proces numer 1. PID: 12407, PPID: 12406
Proces numer 2. PID: 12408, PPID: 12406
Proces numer 3. PID: 12409, PPID: 12406
./a.out
PROCES RODZICIELSKI: 12410
Podaj n: 5
Proces numer 1. PID: 12411, PPID: 12410
Proces numer 2. PID: 12412, PPID: 12410
Proces numer 3. PID: 12413, PPID: 12410
Proces numer 4. PID: 12414, PPID: 12410
Proces numer 5. PID: 12415, PPID: 12410
```

Jeśli terminal tekstowy, w którym są wykonywane jakieś procesy, zostanie wyłączony, przez domniemanie wszystkie te procesy zostaną w tym momencie unicestwione. Jeśli użytkownik chce, aby pomimo zamknięcia terminala procesy te dalej się wykonywały (jako procesy bez terminala sterującego), może to osiągać na kilka sposobów. Sposoby te sprowadzają się do ustawienia pułapki (**trap**) na sygnały wysyłane do procesów w chwili zamykania terminala:

1. **SIGHUP** - jest sygnałem wysyłanym w przypadku kończenia sesji pracy przez podanie polecenia exit.
2. **SIGTERM** - jest sygnałem generowanym w przypadku usunięcia terminala (zamknięcie poprzez przycisk x).
3. itd ...

UWAGA: Identyfikatory sygnałów mogą być podawane zarówno w formie liczbowej jak i mnemonicznej (na przykład: SIGKILL == 9).

Zadanie 4 (0.2 pkt)**.** Na serwerze **SIGMA** przeczytaj opis polecenia **nohup**. Uruchom przy jego użyciu (w tle) skrypt z zadania pierwszego (możesz go sobie przesłać poleceniem **scp**). W jakim przypadku proces będzie nadal działał. Sprawdź zawartość pliku, który utworzył się po użyciu polecenia **nohup**.
Przy użyciu polecenia **trap** uzyskaj ten sam efekt, co przy użyciu polecenia **nohup**.
UWAGA: Po wykonaniu zadania usuń ten proces wraz z plikem, który utworzył się wraz z użyciem polecenia **nohup**.

Zadanie 5 (0.1 pkt)**.** Utwórz skrypt w Bash'u, który będzie wykonywany w tle, będzie odporny na wylogowywanie się właściciela i zamknięcie terminala tekstowego i który co pół minuty będzie wysyłał komunikat Lubię systemy operacyjne!. Zadanie należy wykonać na serwerze **SIGMA**.
Wskazówka: Należy nadpisać sygnały odpowiadające za dane sytuacje.

Zadanie 6 (0.1 pkt)**.** Utwórz skrypt w Bash'u, który będzie wykonywał pętlę nieskończoną, a w momencie przerwania tego procesu (poprzez kombinacje klawiszy: ctrl+c) wyświetli komunikat: Nie ładnie że mnie przerwałeś! Tak łatwo się mnie nie pozbędziesz! Jak wówczas unicestwić proces?
Wskazówka: Należy zastąpić odpowiedny sygnał tym komunikatem.