Laboratorium 3 – Procesy i sygnały

Programy do wykorzystania: `lab3.tar.gz`

Zadanie 1. Procesy - użycie funkcji system()
Uzupełnij/zmodyfikuj kod programu `system.c` tak, aby wyświetlał on informację o dniu tygodnia, w którym jest uruchomiony. Przykład uruchomienia programu w sobotę:

```
Today is not Monday.
Today is not Tuesday.
Today is not Wednesday.
Today is not Thursday.
Today is not Friday.
Today is Saturday.
Today is not Sunday.
```

Uwaga: W programie zakłada się, że data jest wyświetlana w języku angielskim. Może być wymagane ustawienie zmiennej `export LANG=en_EN`. Wykorzystaj polecenie `date` i `grep`.

Zadanie 2. Procesy - użycie funkcji fork()
Napisz program, który tworzy jeden proces potomny. Proces potomny wyświetla *n* razy zestaw małych liter alfabetu (czyli litery od a do z), proces macierzysty wyświetla *n* razy zestaw wielkich liter alfabetu (czyli litery od A do Z. Wartość *n* jest podawana w wierszu wywołania programu. Sprawdź, czy każde wykonanie programu będzie przebiegało tak samo? Spróbuj uruchomić program na różnych komputerach, na przykład na `oceanic`. Wybierz duże *n*.
Uwaga: Nie wolno wykorzystywać liczbowych kodów ASCII!

Zadanie 3. Procesy -użycie funkcji fork(), exec(), wait()

a) Napisz program, który tworzy jeden proces potomny. Proces potomny ma wyświetlić tekst: „`Today is:`” Do wyświetlenia tekstu użyj polecenia `/bin/echo`. Proces macierzysty ma wyświetlić datę. Użyj do tego polecenia `/bin/date`. Przykład:
```
Today is:
Saturday, December 5, 2015
```
b) Wykonaj kilka razy program. Czy zawsze tekst pojawi się przed datą? Zmodyfikuj program tak, aby zawsze tekst pojawiał się przed datą.

Zadanie 4. Kody zakończenia procesów potomnych
W pliku `shell.c` znajduje się prosty shell umożliwiajmy wykonywanie poleceń podanych w wierszu wywołania. Uzupełnij go o zwracanie informacji o sposobie zakończenia wykonywanego polecenia.

Zadanie 5. Obsługa sygnałów

a) Zapoznaj się z programem `sigcatch.c`. Jakie jest zadanie funkcji `signal()` oraz `pause()`? Jak program zachowa się w przypadku otrzymana sygnału INT a jak w przypadku innego sygnału, np. QUIT, TERM?
b) Zapoznaj się z programem `raise.c`. Jakie jest zadanie funkcji `signal()`, `pause()`, `kill()`?

Zadanie 6. Obsługa sygnałów
Napisz program, który co n sekund wypisuje na ekranie komunikat: "Działam dalej". Naciśnięcie klawiszy CTRL+c (sygnał INT) wypisuje napis "Przechwycilem sygnal" i program kontynuuje działanie. Jakiej funkcji użyjesz do odmierzania czasu? Jak zakończyć ten program?

Zadanie 7. Obsługa sygnałów
Napisz program, który prosi o podanie hasła i kończy działanie, jeśli użytkownik nie wprowadzi hasła w określonym czasie. Wskazówka: użyj funkcji alarm().

Zadanie 8 (3pkt).
Napisz program, który wyświetli imiona i nazwiska wszystkich użytkowników, którzy uruchomili aplikację o nazwie podanej w wierszu wywołania. Jeżeli aplikacja nie działa w systemie, wyświetli komunikat: „Aplikacja … nie jest uruchomiona".

Zadanie 9(2pkt).
Tak, jak zadanie 8 z tym, że w pliku o podanej ścieżce przekazujemy listę nazw aplikacji.

Zadania domowe

Zadanie 10.
Napisz program, który pokazuje jak najwięcej atrybutów procesu (identyfikatory, bieżący katalog, katalog główny, maskę plików, ustalenia dotyczące obsługi sygnałów, otwarte pliki, inne).

Zadanie 11.
Napisz program, który pokazuje, które atrybuty procesu macierzystego są dziedziczone przez proces potomny uruchomiony za pomocą funkcji fork(), które zaś otrzymują nową wartość.

Zadanie 12.
Napisz program, który pokazuje, które atrybuty procesu są zachowane przez proces po wykonaniu funkcji exec().

Zadanie 13.
Napisz program, który wyświetla identyfikator procesu (PID) i nazwę związanego z nim polecenia dla wszystkich procesów uruchomionych przez użytkownika podanego w wierszu wywołania programu. Wskazówka: informacje te można uzyskać przeglądając katalog /proc z plików (interesują nas katalogi, których właścicielem jest dany użytkownik) i pliki /proc/PID/status.

Zadanie 14 (dodatkowe 4 punkty)
Napisz aplikację, która wykaże, że sygnały nie są kolejkowane. Aplikacja składa się z dwóch programów:

a) Programu wysyłającego sygnały: sig_sender. Program ten uruchamiany jest z następującymi argumentami: PID procesu, do którego wysyłany będzie sygnał, numer wysyłanego sygnału, liczba wysłanych sygnałów, sygnał kończący proces, do którego wysyłane są sygnały.

b) Programu odbierającego i zliczającego sygnały: sig_receiver. Program ten uruchamiany jest z argumentem określającym liczbę sekund, przez którą należy blokować sygnały.

Przykład wywołania:

```
$ sig_receiver 15 & // blokuj sygnały przez 15 sekund
$ sig_sender 4567 1000000 10 2
sig_receiver: sygnał 10 wyslano 1000000 razy
sig_receiver: sygnal 10 odebrano 65 razy
```