Systemy operacyjne

Laboratorium 8 (Procesy i sygnały cz. 2 - komunikacja międzyprocesowa)

Tutaj znajduje się plik ojciec.sh.
Tutaj znajduje się plik syn.sh.
Tutaj znajduje się plik fifo.c.

Para poleceń **exit** oraz **wait** stanowi najprostsze narzędzie komunikacji między-procesorowej (co było testowane na poprzednim laboratorium). Polecenie o postaci **exit n**, gdzie n jest liczbą naturalną jednobajtową powoduje zakończenie działania procesu i przekazania wartości n jako jego kodu wyjścia. Polecenie o postaci **wait PID** powoduje zsynchronizowanie się (zaczekanie) procesu rodzicielskiego z zakończeniem procesu potomnego o podanym identyfikatorze (PID), odczytanie jego kodu wyjścia i zwrócenie tego kody jako wartości zwracanej przez polecenie.

Zadanie 1 (0.5 pkt)**.** Zaprojektuj analogicznie do pokazanego przykładu parę skryptów, która będzie obliczała dowolną funkcję rekurencyjną.

Zadanie 2 (0.5 pkt)**.** Napisz w języku **ANSI C** program, który utworzy proces rodzicielski i oraz potomny. Następnie proces rodzicielski przekaże procesowi potomnemu liczbę całkowitą, a proces potomny obliczy jakąś dowolną funkcję rekurencyjną. Zastanów się w jakim momencie musi nastąpić przekazanie danych. Dlaczego tak jest? Odpowiedź uzasadnij.

Sygnały służą do przekazywania do wykonywanych procesów powiadomień o charakterze asynchronicznym (nieoczekiwanym przez dany proces). Niektóre z nich mogą być ignorowane przez procesy, mogą powodować ich zawieszenie bądź zakończenie. Jeżeli sygnał nie jest sygnałem nieprzechwytywalnym, procesy mają możliwość zmiany domniemanej reakcji na sygnał poprzez jego przechwycenie (polecenie trap z poprzedniego laboratorium).

Zadanie 3 (0.5 pkt)**.** Poczytaj i znajdź jakie sygnały są nieprzechwytywalne. Następnie utwórz trzy skrypty, którego składnia jest następująca:

```
sygnal.sh
#!/bin/bash
  trap './dostalem' USR1
  trap './koniec' USR2
  while true
  do
    echo "Czekam"
    sleep 5
  done
  exit 0
```

Gdzie **dostałem** oraz **koniec** to skrypty, które:

1. **dostalem** wyświetla wiadomość że otrzymał sygnał **USR1**.
2. **koniec** wyświetla wiadomość że otrzymał sygnał **USR2** i kończy działanie skryptu.

Napisz skrypty **dostalem** i **koniec**, a następnie przetestuj działanie **skrypt.sh**, wysyłając do niego sygnały **USR1** oraz **USR2**.
Wskazówka:
W skrypcie **koniec**, należy zakończyć działanie procesu rodzicielskiego!

Łącza nazwane (kolejki FIFO) są plikami specjalnymi używanymi w komunikacji międzyprocesorowej do przekazywania strumieni bajtów o nieograniczonej długości (kończonych przez naciśnięcie ctrl+d). Są tak zwanymi plikami nietrwałymi (każdy odczyt z łącza powoduje automatyczne usunięcie odczytanych bajtów z łącza) i mają własność synchronizacji procesu zapisującego i odczytującego.

Zadanie 4 (0.5 pkt)**.** Zmodyfikuj skrypty z zadania pierwszego tak, aby przekazanie liczby odbyło się poprzez łącze nazwane, czyli używając polecenia **mkfifo**. Zastanów się jak ma wyglądać przesyłanie liczby do łącza.

ZADANIA DOMOWE: (Do następnych zajęć)
Potoki są łatwe zarówno w tworzeniu, jak i w użyciu. Istnieją dwa rodzaje potoków: potoki nienazwane, wykorzystywane w komunikacji pomiędzy procesem macierzystym, a procesem potomnym, oraz łącza nazwane (kolejki FIFO). Potok nienazwany możemy utworzyć w kodzie procesu za pomocą wywołania systemowego **pipe**, które zwraca parę deskryptorów plików.

Zadanie 5 (1 pkt)**.** Przetestuj działanie programu w języku **ANSI C**, używający potoki nienazwane. Następnie na podstawie jego napisz program, w którym proces macierzysty (po instrukcji fork()) będzie wczytywał liczbę i zapisywał ją do odpowiedniego deskryptora, a proces potomny pobierze tą liczbę, obliczy jakąś funkcję arytmetyczną i zapisze wynik w odpowiednim deskryptorze. Następnie proces macierzysty pobierze wynik i wyświetli go na ekranie.
Wskazówka:
Należy użyć funkcji **wait()**.