KOMUNIKACJA MIĘDZYPROCESOWA

Para poleceń exit i wait stanowi najprostsze narzędzie komunikacji międzyprocesowej. Polecenie o postaci exit n , gdzie n jest liczbą naturalną jednobajtową (tj. z zakresu od 0 do 255) powoduje zakończenie działania procesu i przekazanie wartości n jako jego kodu wyjścia. Polecenie o postaci wait PID powoduje zsynchronizowanie się (zaczekanie) procesu rodzicielskiego z zakończeniem procesu potomnego o podanym identyfikatorze, odczytanie jego kodu wyjścia i zwrócenie tego kodu jako wartości zwracanej przez polecenie.

Użyte symbole specjalne bash'a:

$! - PID ostatnio uruchomionego procesu drugoplanowego
$? - wartość zwrócona przez ostatnio wykonane polecenie pierwszoplanowe

---

1. Przy użyciu poniższej pary skryptów wypróbować możliwość przekazywania niedużej (jednobajtowej) liczby naturalnej od procesu potomnego do procesu rodzicielskiego:

   ```
   ------
   ojciec
   ------

   clear
   echo -n "Podaj liczbe naturalna: "
   read x
   syn $x &
   ojc_wynik=$((x*x))
   wait $!
   syn_wynik=$?
   echo "Wynik obliczenia: $((ojc_wynik+syn_wynik))"
   exit 0

   ---
   syn
   ---

   y=$1
   exit $((2*y))
   ```

   Uruchomić skrypt ojciec (który następnie uruchamia współbieżnie wykonywany skrypt syn), podać daną (nie przekraczającą 6) i odczytać wynik.

   Sprawdzić, że realizowaną funkcją jest x*x+2*x (przy warunku, że nie jest przekroczony zakres obliczeniowy dla liczb jednobajtowych).

   UWAGA: w przypadku nieposiadania ustawionej ścieżki do katalogu bieżącego należy skrypt syn wywoływać w skrypcie ojciec pisząc:

   ```
   ./syn $x &
   ```

Łącza nazwane (kolejki FIFO) są plikami specjalnymi używanymi w komunikacji międzyprocesowej do przekazywania strumieni bajtów o nieograniczonej długości (kończonych przez naciśnięcie ctrl-d).

Są tak zwanymi plikami nietrwałymi (każdy odczyt z łącza powoduje automatycznie usunięcie odczytanych bajtów z łącza) i mają własność synchronizacji procesu zapisującego i odczytującego.

---

2. Utworzyć dwa łącza nazwane przy użyciu poleceń o postaci

   mkfifo nazwa

   Przy użyciu polecenia ls -l sprawdzić atrybuty powstałych plików specjalnych. Otworzyć drugie okno tekstowe, wypróbować możliwość dwustronnej komunikacji pomiędzy dwoma oknami:

(w jednym oknie)	(w drugim oknie)
cat < nazwa1 &	cat < nazwa2 &
cat > nazwa2	cat > nazwa1

   (teraz możemy wpisywać teksty dowolnej długości - na przemian w jednym oknie i w drugim, komunikację kończymy naciskając ctrl-d w każdym z terminali).

   UWAGA: ćwiczenie wykonać na serwerze sigma (po zalogowaniu się przez ssh).

3. Nadać łączom (oraz katalogom, w których są umieszczone) odpowiednie prawa dostępu i w zespołach dwuosobowych powtórzyć eksperyment z poprzedniego zadania dla przypadku dwóch okien tekstowych należących do dwóch różnych użytkowników.

   UWAGA: podobnie, jak w przypadku plików zwykłych, prawa dostępu do łącz zmieniamy przy użyciu polecenia chmod, a usuwamy łącza przy użyciu polecenia rm .

Sygnały służą do przekazywania do wykonywanych procesów powiadomień o zajściu zdarzeń asynchronicznych (nieoczekiwanych przez dany proces). W zależności od swojego charakteru (dokładniej: od numeru sygnału) sygnały przez domniemanie mogą być ignorowane przez procesy, mogą powodować ich zawieszenie bądź zakończenie.

Jeżeli sygnał nie jest sygnałem nieprzechwytywalnym (takim, jak SIGKILL), procesy mają możliwość zmiany domniemanej reakcji na sygnał poprzez jego przechwycenie. Polega to na ustawieniu pułapki (trap) na dany rodzaj sygnału. Pułapka powoduje, że zamiast reakcji domniemanej w chwili otrzymania sygnału proces wykona polecenie podane w poleceniu trap.

---

4. Utworzyć 3 skrypty o następujących nazwach i zawartościach:

   ---
   odbsygnal

```
trap dostalem USR1
trap koniec USR2
while true
do
```

```
   sleep 1
done
exit 0

--------
dostalem
--------

echo "Otrzymalem sygnal USR1 !"
exit 0

------
koniec
------

echo "Koncze dzialanie !"
kill -9 $PPID
exit 0
```

W jednym oknie tekstowym uruchomić skrypt odbsygnal, w drugim sprawdzić PID jego procesu, a następnie wysyłać sygnały przy użyciu poleceń:

```
kill -s USR1 ..... #PID procesu
kill -s USR2 ..... #PID procesu
```

W pierwszym oknie zaobserwować reakcję wykonywanego skryptu na otrzymywane (asynchronicznie) sygnały.

Poniższe zadanie zaleca samodzielne zaprojektowanie i utworzenie prostego protokołu komunikacyjnego do zastosowań wewnątrzsystemowych. Oprogramowanie protokołu komunikacyjnego składa się z programu serwera i programu klienta.

---

5. Utworzyć parę skryptów klient-serwer, z których każdy tworzy łącze o ustalonej nazwie w swoim katalogu. Klient pobiera od swojego użytkownika jedną liczbę naturalną i wpisuje do łącza serwera komunikat złożony z tej liczby oraz z adresu zwrotnego (pełnej nazwy ścieżkowej swojego łącza). Serwer oblicza jakąś prostą funkcję arytmetyczną od otrzymanej wartości liczbowej i wynik obliczenia umieszcza w łączu klienta (pod otrzymanym adresem zwrotnym). Klient pobiera wynik otrzymany od serwera, wyświetla, usuwa swoje łącze i kończy działanie. Serwer powinien działać w pętli nieskończonej i po obsłużeniu poprzedniego klienta powinien czekać na zgłoszenie następnego.

   Uruchomić oba skrypty (serwer jako pierwszy) w oddzielnych oknach tekstowych, wypróbować działanie (skrypt klienta uruchomić kilkakrotnie, sprawdzić prawidłowość wyświetlanych wyników). Przydzielić prawa dostępu tak, aby z usługi serwera mogli korzystać też inni uczestnicy zajęć.

   UWAGA: oba skrypty wykonywać na serwerze sigma (po zalogowaniu się przez ssh).
