Laboratorium 7. Potok

Pobierz plik lab7.tar.gz i rozpakuj go.

Zadanie 1.
Zapoznaj się z programem `pipedemo.c`.
Napisz program, który będzie wysyłał list automatycznie generowany przez program. Do wysyłania listu użyj polecenia mail.

Użyteczna funkcja: do utworzenia składowych polecenia można użyć instrukcji:
```
sprintf(arg,"-s 'Wiadomosc od procesu PID %d'", getpid());
```

Zadanie 2.
Napisz program, składający się z procesu macierzystego i potomnego, w którym

- proces macierzysty generuje komunikat,
- proces potomny odbiera komunikat za pomocą potoku, zamienia go na pisany wielkimi literami i przesyła z powrotem za pomocą potoku do procesu macierzystego
- proces macierzysty drukuje otrzymany komunikat.

Uwaga: należy utworzyć dwa potoki łączące proces macierzysty i potomny.

Zadanie 3.
Zapoznaj się z programem `simpleredirect.c`. Program ten jest równoważny wykonaniu polecenia:
`ls –l | sort –n +4`.

a) Czy ma znaczenie, w którym procesie będzie wykonywane polecenie `ls`?
b) Co będzie się działo, jeśli deskryptory `fd[0]` i `fd[1]` nie będą zamknięte przed wywołaniem `execl`?
c) Rozbuduj program `simpleredirect.c` tak, aby polecenia do wykonania były pobierane z wiersza wywołania programu.

Zadanie 4.
Napisz program, który zbiera komunikaty od wielu programów i wyświetla je na ekranie. Do komunikacji użyj potok nazwany. Wskazówka: Utwórz program rdfifo, którego zadaniem jest utworzenie kolejki FIFO i czytanie z niej danych. Utwórz program wrfifo, który otwiera kolejkę FIFO tylko do zapisu i wpisuje do niej dane (np. swoj pid i czas). W jaki sposób przekażesz wspólną nazwę kolejki FIFO do tych programów? W jaki sposób zapewnić działanie programu zbierającego komunikaty również wtedy, kiedy nie ma programu piszącego do łącza? Jak zapewnić to, że komunikaty pochodzące od różnych programów wyświetlane są w całości, tzn. nie są rozdzielane komunikatami od innych programów?

Zadanie domowe

1. Rozbuduj program z zadania 3 tak, aby potok mógł być dowolnie duży. Przykład: `simpleredirect who sort head`. Jest to odpowiednik polecenia `who | sort | head`.

2. Dodaj możliwość wpisywania poleceń z opcjami.

3. Dokończ zadanie 4.

4. Napisz uproszczoną wersję shell'a który będzie:

- obsługiwał proste polecenia wewnętrzne (
  - **echo**: wyświetlenie argumentów,
  - **cd**: zmiana katalogu bieżącego,
  - **pwd**: wyświetlenie nazwy katalogu bieżącego,
  - **exit:** zakończenie shella,
  - **type:** rozpoznawanie typu polecenia: wewnętrzne/zewnętrzne/nieznane,
  - **file:** rozpoznawanie typu pliku.
- obsługiwał polecenia zewnętrzne (korzystając ze zmiennej PATH),
- polecenia mogą mieć opcje i argumenty
- ignorował sygnały: **SIGINT** (**Ctrl+c -** podawany z klawiatury**)** i **SIGQUIT (Ctrl+\ -** podawany z klawiatury**).**
- obsługiwał przeadresowanie we (<) i wy (>, >>)
- obsługiwał potok (dowolna liczba poleceń)