Potoki nazwane i nienazwane

Zadanie 1 (40%)

Napisz interpreter poleceń przechowywanych w pliku. Ścieżka do pliku to pierwszy argument wywołania programu.

Polecenia w pliku przechowywane w następującej postaci:

```
# Definicja składników
składnik_1 = prog_1 arg_1 ... argn_1 | prog_2 arg_1 ... argn_2 | ... | prog_N arg_1 ... argn_N
składnik_2 = prog_1 arg_1 ... argn_1 | prog_2 arg_1 ... argn_2 | ... | prog_M arg_1 ... argn_M
...
składnik_K = prog_1 arg_1 ... argn_1 | prog_2 arg_1 ... argn_2 | ... | prog_Z arg_1 ... argn_Z
```

```
# Wykonanie potoku
składnik_1 | składnik_2 | składnik_K
```

Przykład

```
składnik1 = cat /etc/passwd | wc -l
składnik2 = ls | grep '^a'
składnik3 = grep 11

składnik1 | składnik3 # ⇔ cat /etc/passwd | wc -l | grep 11
składnik1             # ⇔ cat /etc/passwd | wc -l
składnik2             # ⇔ ls | grep '^a'
```

- Interpreter powinien uruchomić wszystkie polecenia w osobnych procesach, zapewniając przy użyciu potoków nienazwanych oraz funkcji `dup2()`, by wyjście standardowe procesu k było przekierowane do wejścia standardowego procesu (k+1)
- Można założyć ograniczenie górne na ilość obsługiwanych argumentów oraz ilość połączonych komend w pojedynczym poleceniu (co najmniej 3).
- Po uruchomieniu ciągu programów składających się na pojedyncze polecenie (linijkę) interpreter powinien oczekiwać na zakończenie wszystkich tych programów.
  Program należy zaimplementować, korzystając z funkcji: `pipe()`, `fork()` oraz `exec()`.

Zadanie 2 (20%)

Napisać program przyjmujący jeden (`nadawca` lub `data`) lub trzy argumenty (`<adresEmail> <tytuł> <treść>`):

- W przypadku wywołania z jednym argumentem uruchamiany jest (za pomocą `popen()`) program *mail*. Program użytkownika ma wypisywać listę e-maili posortowaną alfabetycznie wg. adresów e-mail (argument `nadawca`) lub wg. daty otrzymania e-maili (argument `data`)
- Jeżeli program zostanie wywołany z trzema argumentami, to (za pomocą `popen()`) uruchamiany jest program *mail* i za jego pomocą wysyłany jest e-mail do określonego nadawcy z określonym tematem i treścią

Zadanie 3 (40%)

W problemie producenta i konsumenta występują dwa rodzaje procesów, które dzielą wspólny bufor dla produkowanych i konsumowanych jednostek. Zadaniem producenta jest wytworzenie surowca, umieszczenie go w buforze i rozpoczęcie pracy od nowa. Konsument pobiera surowiec z bufora i wykorzystuje go.

30%

Przy pomocy potoków nazwanych zaimplementować problem Producenta i Konsumenta. Napisać dwa niezależne programy - Producent oraz Konsument, które będą komunikować się poprzez potok nazwany (kolejkę FIFO). Do potoku pisać będzie wiele procesów wykonujących program Producenta, a czytał będzie z niej jeden proces Konsumenta. Dla zademonstrowania, że nie doszło do utraty ani zwielokrotnienia towaru surowiec będzie pobierany z pliku przez Producenta (każdy Producent czyta dane z osobnego pliku) i umieszczany w innym pliku przez Konsumenta (**otrzymane dane od producenta nr *i* mają się pojawić w linii nr *i* pliku wynikowego**).
Producent:

- przyjmuje cztery argumenty: ścieżka do potoku nazwanego, numer wiersza, ścieżka do pliku tekstowego z dowolną zawartością, *N* - liczba znaków odczytywanych jednorazowo z pliku
- otwiera potok nazwany
- wielokrotnie (aż do odczytania całego pliku):
  - odczekuje losową ilość czasu (np. 1-2 sekund)

    - zapisuje do potoku nazwanego: numer wiersza oraz odczytany fragment pliku (*N* odczytanych znaków)

Konsument:

- przyjmuje trzy argumenty: ścieżka do potoku nazwanego, ścieżka do pliku tekstowego (do którego będzie zapisywany odczytany tekst), *N* — liczba znaków odczytywanych jednorazowo z pliku
- otwiera potok nazwany
- wielokrotnie:
    - odczytuje numer wiersza *i* oraz *N* kolejnych znaków potoku nazwanego
    - umieszcza odczytane znaki w linii nr *i* pliku tekstowego (różnym od plików, z których korzystają producenci)

Pliki tekstowe powinny być krótkie (na 5-10 odczytów) i umożliwiać sprawdzenie poprawności działania (brak utraty, zwielokrotnienia surowca). W szczególności każdy Producent powinien otrzymać wygenerowany w dowolny sposób plik tekstowy z dowolną zawartością, ale w istotny sposób różniącą się od zawartości plików innych Producentów. Na przykład jeden producent może otrzymać plik zawierający tylko konkretną literę, inny tylko liczby itd.

10%

Utwórz plik *wnioski.txt* zawierający wyniki (wraz z wnioskami) następujących sprawdzań:

- Sprawdzić, że potoki nazwane działają dla niezależnych procesów — utworzyć potok z linii komend, a następnie uruchomić Producenta i Konsumenta w różnych terminalach. Dodatkowo należy napisać program, który tworzy potok nazwany, a następnie uruchamia program Konsumenta i pięciu Producentów (z różnymi argumentami).
- Sprawdź, dla trzech istotnie różnych wartości *N* (np. dla *N* = 5, *N* > *PIPE_BUF*), następujące przypadki:
    1. wielu producentów, jeden konsument
    2. jeden producenta, wielu konsumentów
    3. wielu producentów, wielu konsumentów

  Uwagi:
    1. Ponieważ, w tym przypadku kilka procesów będzie zapisywać do jednego, wspólnego pliku, dlatego należy użyć funkcji `flock()`
    2. Należy napisać program lub skrypt sprawdzający, czy zawartość pliku wejściowego, w całości, pojawiła się w odpowiedniej linii pliku wynikowego
    3. Uruchamianie powyższych przypadków testowych oraz programu/skryptu sprawdzającego ma się odbywać za pomocą komendy `make test`