Systemy operacyjne
Laboratorium 6
Przemysław Ziaja (303187)

ZAD 1. - PIPE, WRITE, READ, CLOSE
Pipe - kontener, który odpowiada za przekazywanie danych pomiędzy procesami. W działaniu jest podobny do kolejki. Dane przez jeden proces są zapisywane do pipe'a, a drugi wyciąga je z pipe'a.
Dane są składowane w RAMie. Mają ograniczony rozmiar, w moim przypadku są to 4kb. Rozmiarem można manipulować przy pomocy funkcji fcntl.
Warto nadmienić, że do pipe'a może być utworzonych wiele deskryptorów do zapisu i do odczytu. Jeżeli wszystkie deskryptory do pipe'a są zamknięte to pipe jest kasowany. Jednakże najlepiej mieć otwarty tylko jeden deskryptor do zapisu i odczytu pomiędzy procesami. Sam długo siedziałem nad debugowaniem programu i powodem problemów były otwarte pipe'y w różnych procesach.

Write - funkcja do zapisu. Przyjmuje deskryptor do którego mają zostać zapisane dane, pointer char, z którego mają być pobierane znaki do zapisu i ilość znaków, które mają być zapisane.

Read - funkcja do odczytu. Przyjmuje deskryptor z którego odczytuje znaki, pointer char do którego mają zostać zapisane odczytane znaki oraz ile znaków może odczytać (rozmiar bufora). Osobiście wygodniejsze wydaje mi się używanie funkcji getline, zasady działania podobne, a wygodniej operuje się na liniach niż na blokach tekstu.

Close - zamykanie deskryptora pliku.

ZAD 2. - DUP DUP2
DUP - kopiuje deskryptor pliku. Skopiowany deskryptor przypisuje do najniższego pustego deskryptora. Przykład:

```
INT CP_DESC = DUP(DESC);
```

DUP2 - to samo co DUP tylko pozwala na wybranie deskryptora, do którego ma zostać zapisana kopia. Przykład przypisuję DESC jako wyjście programu, tzn mogę rezultat programu, który normalnie jest wypisywany na ekranie zapisać do pliku:

```
DUP2(DESC,1);
```

PODSUMOWANIE
Bardzo wygodne w użyciu jest funkcja DUP2. Przekierowanie wyjścia z programu do pipe przyspiesza pracę. Nie udało niestety mi się zapisywać danych do pipe'a przy użyciu dup2 i printf, dlatego zostawiłem funkcję write.
W zadaniu 3.1 dużo błędów w działaniu programu było spowodowanych pozostawieniem otwartych deskryptorów, których nie używałem. Na początku chciałem chciałem stworzyć 1 pipe, który przesyłałby wyniki działani obu podprocesów. Jednak po późniejszych ustaleniach, zdecydowałem, że bezpieczniej będzie utworzyć dla każdego podprocesu osobnego pipe'a zwracającego dane.