Maryna Lukachyk

308294

Laboratorium 6

Komunikacja międzyprocesowa - łącza nienazwane

1 Tworzenie prostych łączy jednokierunkowych

```
#include <unistd.h>

int pipe(int filedes[2]);
```

pipe tworzy parę sprzężonych deskryptorów pliku, wskazujących na i-węzeł potoku i umieszcza je w tablicy wskazywanej przez *filedes*. *filedes[0]* jest dla odczytu, a *filedes[1]* dla zapisu.

47

The fork makes the both processes have access to both ends of the pipe. This is not desirable.

The reading side is supposed to learn that the writer has finished if it notices an EOF condition. This can only happen if all writing sides are closed. So it is best if it closes its writing FD ASAP.

The writer should close its reading FD just in order not to have too many FDs open and thus reaching a maybe existing limit of open FDs. Besides, if the then only reader dies, the writer gets notified about this by getting a SIGPIPE or at least an EPIPE error (depending on how signals are defined). If there are several readers, the writer cannot detect that "the real one" went away, goes on writing and gets stuck as the writing FD blocks in the hope, the "unused" reader will read something.

So here in detail what happens:

- parent process calls `pipe()` and gets 2 file descriptors: let's call it `rd` and `wr`.
- parent process calls `fork()`. Now both processes have a `rd` and a `wr`.
- Suppose the child process is supposed to be the reader.

  Then

  - the parent should close its reading end (for not wasting FDs and for proper detection of dying reader) and
  - the child must close its writing end (in order to be possible to detect the EOF condition).

2 Praca z łączami komunikacyjnymi

Aby stworzyć kopię deskryptora pliku, używamy funkcji:

int dup(int oldfd);

int dup2(int oldfd, int newfd);

Funkcja dup tworzy kopię deskryptora, która odnosi się do *tego samego* miejsca w tablicy otwartych plików, co oryginał. Jest to zatem zupełnie inna sytuacja niż przy wielokrotnym otwarciu tego samego pliku. Funkcja dup duplikuje deskryptor oldfd i przekazuje wartość nowego deskryptora. Wiadomo przy tym, że przekazywany deskryptor jest deskryptorem o najmniejszej wartości spośród aktualnie dostępnych. Funkcja dup2 zamyka deskryptor newfd, jeśli był on otwarty i duplikuje oldfd na tę właśnie wartość. Wynikiem dup2 jest numer nowego deskryptora (czyli newfd). W wypadku błędu obie funkcje dają w wyniku wartość -1.

File Descriptor Number	Initially	After close of File Descriptor 0	After dup
0	Standard input	{closed}	Pipe file descriptor
1	Standard output	Standard output	Standard output
2	Standard error	Standard error	Standard error
3	Pipe file descriptor	_ipe file descriptor	Pipe file descriptor

3 Ćwiczenia dot. łączy nienazwanych