EXPERIMENT 7 – IPC USING PIPES

OBJECTIVES

Learn to use pipes for inter process communication.

TIME REQUIRED : 2 hrs

PROGRAMMING LANGUAGE : C

SOFTWARE REQUIRED : Ubuntu/Fedora, gcc/gc, Windows, Dev, NetBeans

HARDWARE REQUIRED : Core i5 in Computer Labs

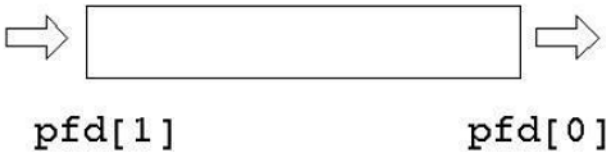
INTER PROCESS COMMUNICATION USING SIMPLE PIPES

Inter- process Communication (IPC) is set of ways to communicate between processes or threads. Pipe is one of the ways to communicate. Pipe is special file; the child inherits the pipe from its parent process. Process writes to one end of the pipe (write-end), other reads from the other end of pipe (read-end). Its unidirectional (one-way communication). For two-way communication we use two pipes.

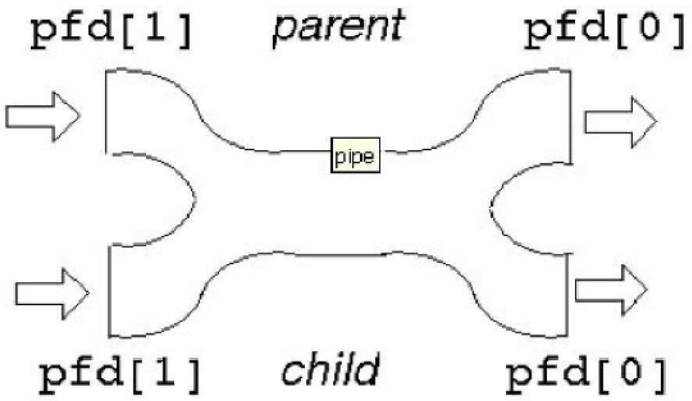
```
int pipe( int fd[2]);
```

Two file descriptors are returned through the argument: fd[0] is open for reading fd[1] is open for writing. Typically, a process creates the pipeline, then uses "fork" to create a child process. Each process now has a copy of the file descriptor array; one process writes data into pipeline, while the other reads from it.

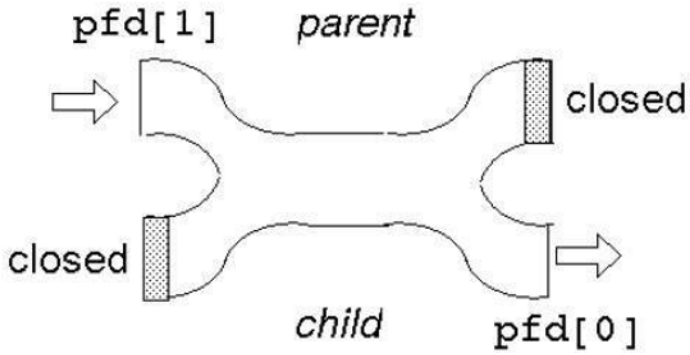
Before fork



After fork



This gives two read ends and two write ends. The read end of the pipe will not be closed until both read ends are closed, and the write end will not be closed until both the write ends are closed. Either process can write into the pipe, or either can read from it. Which process will get what is not known? For predictable behaviour, one of the processes must close its read end, and the other must close its write end. Then it will become a simple pipeline again.



TASK 7.1

Create a file named mypipe.c. Type in the following code:

```
#include<stdio.h>
#include<unistd.h>

int main() {
  int pipefds[2];
  int returnstatus;
  char writemessages[2][20]={"Hi", "Hello"};
  char readmessage[20];
  returnstatus = pipe(pipefds);
  if (returnstatus == -1) {
    printf("Unable to create pipe\n");
    return 1;
  }
  printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);
  write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
  read(pipefds[0], readmessage, sizeof(readmessage));
  printf("Reading from pipe – Message 1 is %s\n", readmessage);
```

```
  printf("Writing to pipe - Message 2 is %s\n", writemessages[0]);
  write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
  read(pipefds[0], readmessage, sizeof(readmessage));
  printf("Reading from pipe – Message 2 is %s\n", readmessage);
  return 0;
}
```

Compile, execute and show outcome her: -