National University of Computer and Emerging Sciences



**Laboratory Manual**

***(Operating Systems)***

|  |  |
| --- | --- |
| Semester | Spring 2018 |

Department of Computer Science

FAST-NU, Lahore

Pipes

Ordinary pipes allow two processes to communicate in standard producer consumer fashion: the producer writes to one end of the pipe (the **write-end)** and the consumer reads from the other end (the **read-end). As a result, ordinary** pipes are unidirectional, allowing only one-way communication. If two-way communication is required, two pipes must be used, with each pipe sending data in a different direction.

**References**: Operating System concepts Page n.o 142 section 3.6.3.1

A pipe has a read end and a write end.

Data written to the write end of a pipe can be read from the read end of the pipe.

**On UNIX and Linux systems, ordinary pipes are** **constructed using the function**

• int pipe(int fd[2]) -- creates a pipe

• returns two file descriptors, fd[0], fd[1].

• fd[0] is the read-end of the pipe

• fd[1] is the write-end.

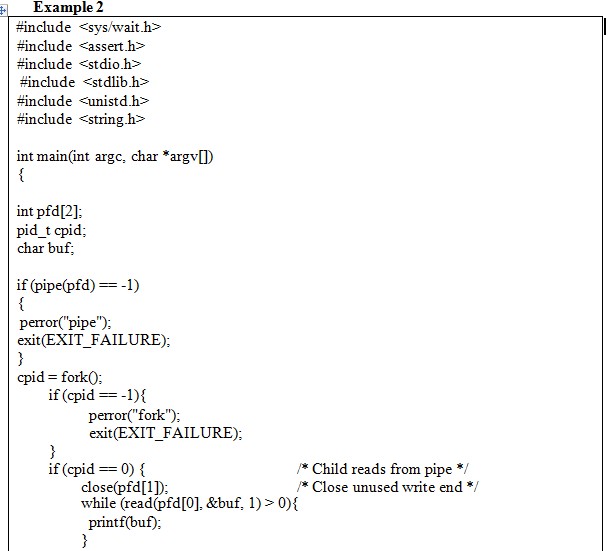
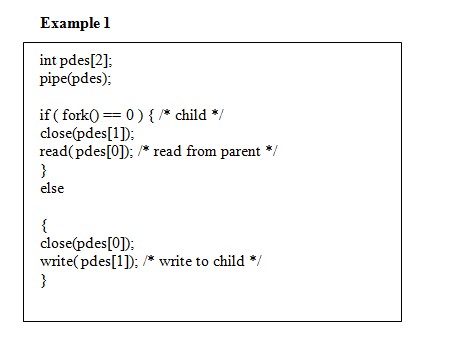
* fd[0] is opened for reading,
* fd[1] for writing. pipe() returns 0 on success, -1 on failure and sets errno accordingly.

• The standard programming model is that after the pipe has been set up, two (or more) cooperative processes will be created by a fork and data will be passed using read() and write().

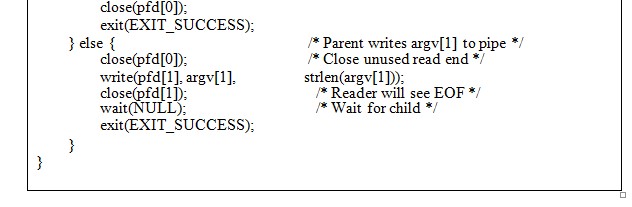
* Pipes opened with pipe() should be closed with

close(int fd).

**Reference**: <http://linux.die.net/man/2/pipe>



Example 2 continues…



**When pipe() System Call Fails:**

The pipe() system call fails for many reasons, including the following:

1 At least two slots are not empty in the FDT—too many files or pipes are open in the process.

2 Buffer space not available in the kernel

Inlab Questions

Q1. Design a program using ordinary pipes in which one process sends a string message to a second process, and the second process reverses the case of each character in the message and sends it back to the first process. For example, if the first process sends the message Hi There, the second process will return hi tHERE. This will require using two pipes, one for sending the original message from the first to the second process, and the other for sending the modified message from the second back to the first process.

Q2. Design a file-copying program named FileCopy using ordinary pipes. This program will be passed two parameters: the first is the name of the file to be copied, and the second is the name of the copied file. The program will then create an ordinary pipe and write the contents of the file to be copied to the pipe. The child process will read this file from the pipe and write it to the destination file. For example, if we invoke the program as follows: FileCopy input.txt copy.txt the file input. txt will be written to the pipe. The child process will read the contents of this file and write it to the destination fil copy. txt.