CS 451 – Operating Systems

Lab 02 Assignment

Due: 11:59 PM on Saturday, 1 August 2020

You are to submit your solutions, to the Lab 2 folder on Blackboard by the due date and time.

Objective

This assignment is intended to provide an introduction to Inter-process communication. You are expected to refer to the textbook and references mentioned below before you start the lab.

Recommended Systems/Software Requirements:

Any flavor of Linux

Tutorial

In computing, a named pipe (also known as a FIFO) is one of the methods for intern-process communication.

- It is an extension to the traditional pipe concept on Unix. A traditional pipe is "unnamed" and lasts only as long as the process.
- A named pipe, however, can last as long as the system is up, beyond the life of the process. It can be deleted if no longer used.
- Usually a named pipe appears as a file, and generally processes attach to it for inter-process communication. A FIFO file is a special kind of file on the local storage which allows two or more processes to communicate with each other by reading/writing to/from this file.
- A FIFO special file is entered into the filesystem by calling mkfifo() in C. Once we have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it.

Name

mkfifo - make a FIFO special file (a named pipe)

Synopsis

```
#include <sys/types.h>#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

Description

mkfifo() makes a FIFO special file with name *pathname*.

mode specifies the FIFO's permissions. It is modified by the process's **umask** in the usual way: the permissions of the created file are **(***mode* **& ~umask)**.

A FIFO special file is similar to a pipe, except that it is created in a different way. Instead of being an anonymous communications channel, a FIFO special file is entered into the file system by calling **mkfifo**().

Once you have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it. Opening a FIFO for reading normally blocks until some other process opens the same FIFO for writing, and vice versa. See ***fifo****(7)* for nonblocking handling of FIFO special files.

Using FIFO

As named pipe (FIFO) is a kind of file, we can use all the system calls associated with it i.e. *open*, *read*, *write*, *close*.

Return Value

On success **mkfifo**() returns 0. In the case of an error, -1 is returned (in which case, *errno* is set appropriately).

Reading From and Writing to a Named Pipe

- Reading from and writing to a named pipe are very similar to reading and writing from or to a normal file.
- The standard C library function calls
  - read() and
  - write()

  can be used for reading from and writing to a named pipe.
- These operations are **blocking**, by default.

Note: This lab is an extension of Lab 1 (In order to finish this lab, you must complete lab 1).

Task 1

Implementing output redirection

Include an additional functionality of output redirection **(>)** to your "**myshell.c**" program you developed in lab 1. When an output redirection is provided, your program should write the output into a file. The filename will be specified by the user.

- If the output file does not exist, you should create it.
- If the output file already exists, you should not overwrite it.

Following are the valid execution commands:

```
$ make
$ ./myshell

myshell> mysort input
4
6
7
10
200

myshell> mysort input > output

information stored in output file

myshell> mysort input > output

output: File exists
```

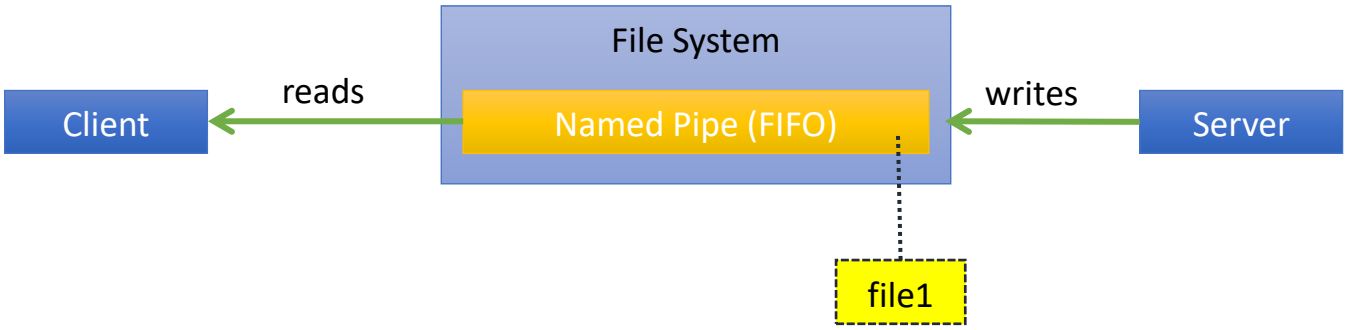
- You will need to use the **dup2** system call and the appropriate constant (**STDOUT_FILENO** or **STDIN_FILENO**) to implement output redirection.
- First open the file (use creat for outfiles) and then use **dup2**.
- 0 is the file descriptor for **STDIN** and 1 is the file descriptor for **STDOUT**. For more information on these calls, consult the man pages.

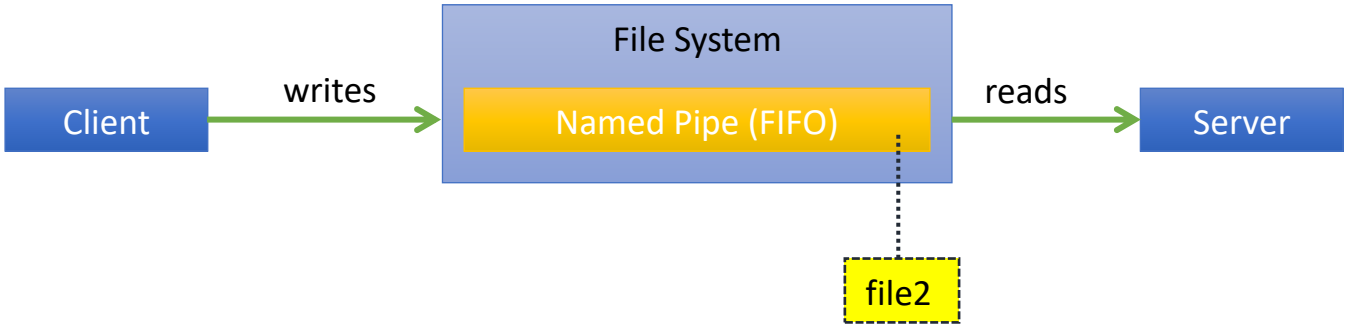
```
dup2(fdFileYouOpened, fileno(stdout))

dup2(fdFileYouOpened, fileno(stdin))
```

Task 2

Develop a chat box using named pipes. You will write two simple programs client.c and server.c that use a named pipe to communicate. The programs will set up a named pipe using **`mkfifo()`**. Your communication should be a full-duplex communication. A sample file is provided in the lab 2 dropbox.

Note: client and server need to be executed in separate shells (or terminals). First start the server and then the client.





Deliverables

For this assignment, you need to hand in **four** distinct items:

- Your source code (**"myshell.c", "mysort.c" "client.c", "server.c"** (no object files or executables, please!)
- A makefile
- A README file with some basic documentation about your code