Systems programming

Week 1 – Lab 2

Client-Server and Pipes

In this laboratory students will implement a simple client-server system, where the client sends to the server commands to be executed.

For communication these processes will use FIFOs, and for the server to get the commands, a dynamic library (like the one on the Lab 1) will be used.

1 FIFOs

fifo(7) - Linux manual page (man7.org)

pipe(7) - Linux manual page (man7.org)

mkfifo(3) - Linux manual page (man7.org)

FIFOs are special files that do not store data in disk (as opposed to regular files) but allow synchronous reading and writing by two processes: every data that is written on the FIFO can be immediately read by another process.

In order to create these “special files”it is necessary to use the **mkfifo** function. After the creation of the fifo, regular file management function (open, read, write, fopen, fread, fwrite) can be used. After a process opens a FIFO for reading (open(…. ,)_RDONLY)), he will be blocked on the read function until some other process writes in the FIFO, and he will immediately read the data other process has written to.

Students should read the provided PDF to further understand how FIFOs work:

The The Linux Programming Interface – Sections 44.1, 44.2, 44.7 and 44.8

1.1 Exercise 1

Observe the two program supplied in the **fifo-example** directory, compile both programs, execute each one in different terminals and answer the following questions:

- what does the **mkfifo** function does? (hint: look ate the /tmp directory)

- What happens if the user takes too long to launch on the **fifo-read** after launching the **fifo-write**?
- What happens if the user takes too long to launch on the **fifo-write** after launching the **fifo-read**?
- What happen during the regular execution of the programs? (when the user type a string of a integer in the fifo-writer)
- What happens if the fifo-read is killed? (kill this program with the Ctl-C)?
- What happens if the fifo-write is killed? (kill this program with the Ctl-C)?

1.2 Exercise 2

Open another window and launch a new **fifo-read**. Type strings and integer in the **fifo-write** and observe what is written be each **fifo-read**. What happens?

2 Exercise 3

open(2) - Linux manual page (man7.org)

read(2) - Linux manual page (man7.org)

write(2) - Linux manual page (man7.org)

Implement a simple client-server system composed of two programs (**client** and **execution-server**) that communicate using a FIFO.

The **execution-server** will load a dynamic library (like in Lab 1) that includes a series of functions. The **execution-server** will read from the FIFO names of functions in a loop, execute them and print the results on the screen.

The function implemented in the dynamic library should not receive any argument and return an integer:

```
int (* func)()
```

The client will read strings from the keyboard and send them to the execution server through the pipe. Each string corresponds to the name of the function to be executed on the **execution-server**.

The server should not read any information from the keyboard.

3 Exercise 4

Modify the previous exercise so that the result of the functions executed in the **execution-server** are printed by the client.

4 Exercise 5

Modify the previous exercise so that it is possible to have function with different interfaces in the library.

The library should now contain function that match:

```
int (* func)()
int (* funct) (int)
```

The client will now **also** read the function name and an integer and send all this information to the execution-server.