School of Computer Science

COMP30640

Lab 7
Inter Process Communitation (IPC): Named Pipes

Teaching Assistant:	Thomas Laurent
Coordinator:	Anthony Ventresque
Date:	Friday 26th October, 2018
Total Number of Pages:	2

Like un-named/anonymous pipes, named pipes provide a form of IPC (Inter-Process Communication). With anonymous pipes, there's one reader and one writer, but that's not required with named pipes, any number of readers and writers may use the pipe.

Named pipes are visible in the filesystem and can be read and written just as other files are:

```
$> ls -l test_pipe
prw-rw-r-- 1 aventresque aventresque 0 Oct 26 15:38 test_pipe
```

You can create named pipes using the following command:

```
$> mkfifo test_pipe
```

which in this case would create a named pipe called `test_pipe`.

1 Playing with your First Named Pipes.

Create a named pipe called `pipe_test`.
Open **two terminals** and run the following commands:

```
$> read something < test_pipe; echo $something
```

in one of the terminals and:

```
$> echo "who are you?" > test_pipe
```

in the other terminal. What happens?

2 Named Pipes in Scripts.

Now create the two scripts below, where the writer writes in the pipe something that is given by the user (using `read` from the terminal) while the reader continuously reads from the pipe and displays it in the terminal.

`write_test.sh`:

```
#!/bin/bash
while true; do
        read input
        echo $input > test_pipe
done
```

`read_test.sh`

```
#!/bin/bash
while true; do
        read input < test_pipe
        echo received from the pipe: $input
done
```

Again, use two terminals to test your scripts. You can stop both scripts with control+c

3 Problem.

Now use **one** single script that reads from one pipe and writes on another pipe. The names of those two pipes should be given as arguments to the script. Try launching two processes of this script (with different parameters) and make them talk to each other (process 1 reads from process 2's output pipe and outputs to the pipe that process 2 reads from). Create 2 pipes and start 2 processes of the script, one in each terminal. In a 3rd terminal, send a message to either one of the two pipes. What happens?

Can you solve this issue by testing whether a message has been received already before (in short do not send back a message that you've received twice in a row)?

4 IPC to Solve Concurrency Issues.

Can you use named pipes to solve the concurrency problem seen in the lab last week? **Hint:** Could we create something to pass between the pipes to tell the process it can run (write.sh)?