Exercises
Input / output

Computadors

Grau en Ciència i Enginyeria de Dades

Xavier Verdú, Xavier Martorell

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2020-2021 Q2

Creative Commons License

Exercise 1

- Is there any performance difference comparing both codes?

CODE A

```
while ((len = read(0, buf, 1)) > 0)
    write(1, buf, len);
```

```
char buf[1024];
```

CODE B

```
while ((len = read(0, buf, 1024)) > 0)
    write(1, buf, len);
```

- And comparing these ones?

```
char buf[1024*1024];
```

```
while ((len = read(0, buf, 1024)) > 0)
    write(1, buf, len);
```

```
while ((len = read(0, buf, 2048)) > 0)
    write(1, buf, len);
```

Exercise 2

- What are we doing with the following code snippet?

```
…
pid = fork();
if (pid == 0){
    close(0);
    open(“/dev/deviceA”, O_RDONLY);
    close(1);
    open(“/dev/deviceB”, O_WRONLY);
    execlp(“./myprog”, “./myprog”, NULL);
}
…
```

Hint: can you write it in a shell command syntax using channels redirections?

Exercise 3

- Comment – advantages and inconveniences - about this code snippet

```
…
pid = fork();
if (pid == 0){
    close(0);
    open(“/dev/deviceA”, O_WRONLY);
    close(1);
    open(“/dev/deviceB”, O_RDWR);
    execlp(“./myprog”, “./myprog”, NULL);
}
…
```

Hint: can you write it in a shell command syntax using channels redirections?

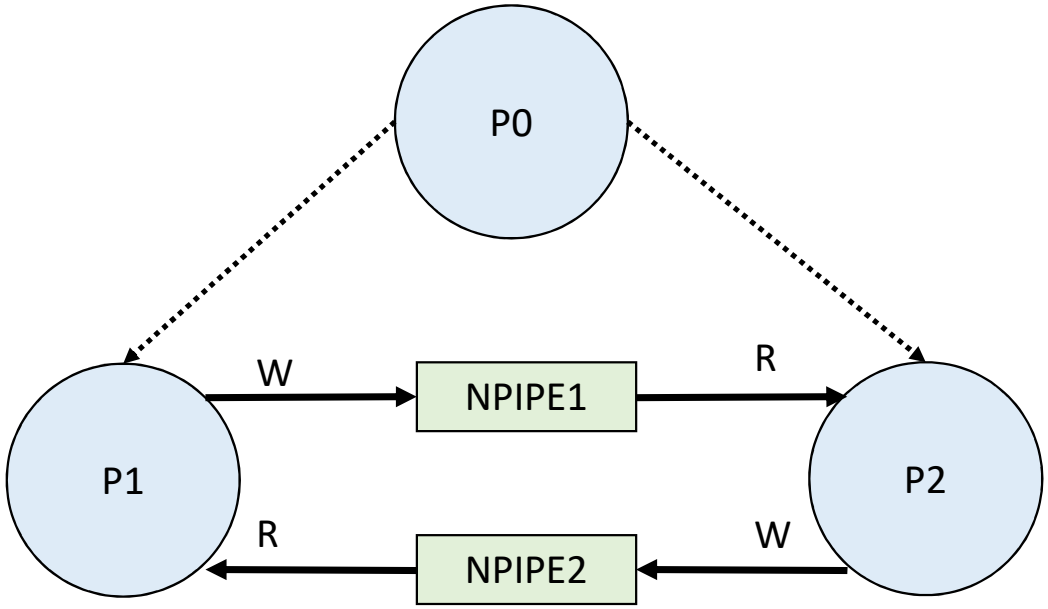
Exercise 4

- Assuming the device /dev/urandom generates pseudorandom bytes, implement a code to:
  - read 100 random bytes from this device
  - write those values to the stdout using integer format (i.e. “int”)
  - write those values to the stderr using ASCII format
    - **HINT:** sprintf(buf,“%d”, num); //converts a number to ASCII format

- Redirect both outputs (stdout and stderr) to different output files
  - Compare both output files with “xxd” command as well as their size

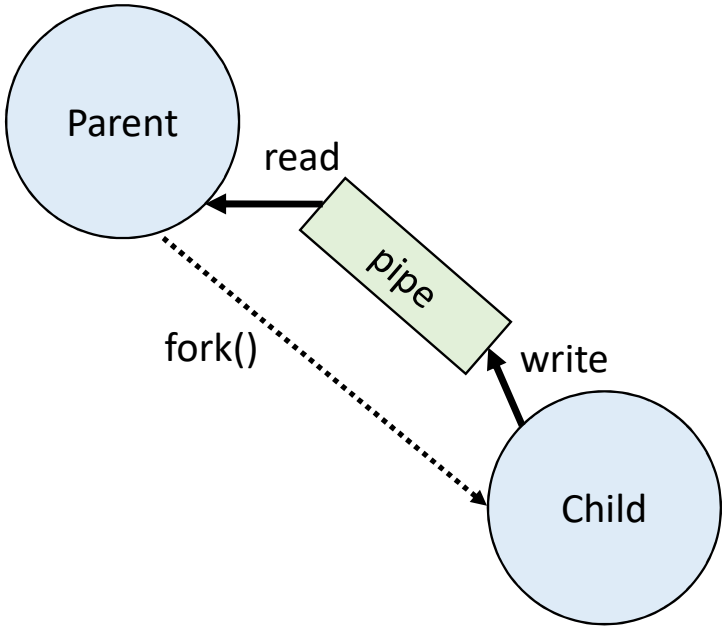
Exercise 5

- Assuming there are two named pipes (“NPIPE1” and “NPIPE2”) implement the following communication diagram



Exercise 6

- Implement the most basic communication of a single message passing through a pipe between (from) a child and (to) its parent process



Exercise 7

- Implement the complete communication schema supporting several messages passing through a pipe between (from) a child and (to) its parent process
  - Get the messages from the command line
  - Parent and Child processes close the non-used pipe descriptors
  - The Child process sends the messages one by one
  - The Parent process receives the messages and displays them on the standard output
  - The Child process closes the output channel of the pipe, and exits This signals an end-of-file to the Parent process
  - The Parent process detects the end-of-file condition and waits for the child