NOVA SCHOOL OF SCIENCE & TECHNOLOGY
DEPARTAMENTO DE INFORMÁTICA

Implementation of a simple Shell (part II)

LAB 3

Objectives

Implementation of command composition via an unnamed pipe within a simple shell.

Command composition using a pipe

Considering the simple shell you implemented in Lab2, extend your code to support a simple pipe linking two separate commands, e.g.,

```
cat ficheiro.txt | sort
```

In this case, the standard output of the command **`cat`** is redirected to the pipe, whereas the standard input of command **`sort`** is redirected to the pipe. The output will show the result of sorting the lines in file *ficheiro.txt*. The resulting process tree should be like:

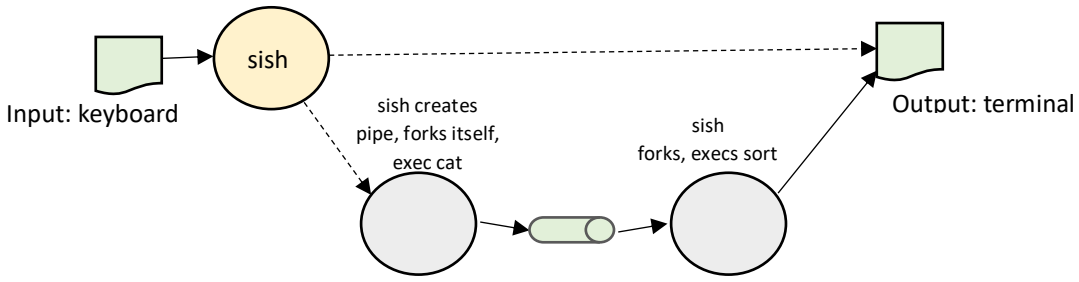

Figure 1. Launching two commands (in gray) communicating via a pipe

Redirecting the standard I/O (optional)

Extend your shell to execute commands with redirected output and/or input to files. For that purpose, the user can write a command line such as

```
command arguments > filename
```

to inform that the standard output of the process running the command should be redirected to the file *filename*. Similarly, a command line such as

```
command arguments < filename
```

must execute the command with its standard input redirected to the file *filename*.

Command composition using multiple pipes (optional)

Extend the solution supporting two commands communicating via a pipe, to support a pipeline execution, i.e. a sequence of multiple commands communicating via pipes. For instance, given a text file of NOVA's students ,named *students.txt*, one per line, the following pipeline

```
cat students.txt | grep LEI | wc -l
```

may count how many students are enrolled in "LEI" (computer science engineering, BSc).

Bibliography

- Section “Laboratory: Tutorial” of recommended book: http://pages.cs.wisc.edu/~remzi/OSTEP/lab-tutorial.pdf
- Must read, from the FSO recommended book, the chapter about process creation available on https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-api.pdf
- On-line manual pages (use man command) for LibC and system call functions: **`fork`**, **`wait`**, **`exit`**, **`execve`**, **`execvp`**, **`strcmp`**, **`open`**, **`close`**, **`pipe`**, **`dup2`**, **`dup`**, **etc**.