Project 1

Deadline 21/02 23:59

Project #1

Implement a simple shell in C

- You will implement a simple shell, which uses exclusively **execv** and **fork** for sub-process.
  - Fork duplicates the process
  - Execv replace the current process by another one

- It must support launching programs with arguments, and return to the prompt when the program terminates, but no other concepts such as variable, substitution, pipes, chaining, …

- It will exit upon typing "exit" or CTRL+D (EOF)

- Command line can be limited to 255 arguments, while each arguments may be limited to 255 characters.

- The first prompt must be "> ",then "RET> " where RET is the return code of the last command. If there was no command when pressing enter, "> " is shown

Example

```
$ gcc -std=gnu99 -o shell shell.c && ./shell
> /bin/ls
shell  shell.c       shell.c~  shell.tar.gz

0> /bin/ls -al
total 32
drwxr-xr-x. 2 tom tom 4096  9 fév 12:23 .
drwxr-xr-x. 3 tom tom 4096  9 fév 11:28 ..
-rwxr-xr-x. 1 tom tom 9032  9 fév 12:23 shell
-rw-r--r--. 1 tom tom 1133  9 fév 12:23 shell.c
-rw-r--r--. 1 tom tom 1134  9 fév 12:23 shell.c~
-rw-r--r--. 1 tom tom  208  9 fév 11:41 shell.tar.gz

0>
```

Terminal

PATH support

- Look in the folders of $PATH (separated by :) in order to find the program to launch
- This should avoid to type "/bin/ls", as bin should be in the path, typing ls should be enough

DO NOT USE exec*p variants that do that themselves, only support it manually

/bin/chsh

Will you dare trust your own shell?

Submission

- You must submit a file named shell.c using the submission platform (https://submit.montefiore.ulg.ac.be)
- Currently, only compile test and a few input/output comparisons
- **Per group of 2** that you will keep the whole semester

Project 1

Supplementary info + Q&A

Project 1 supplementary info

- Your shell must support the ***cd*** built-in command !
- However, *ls* is just a normal command
- No other built-ins than cd are required as of now
  - Tough, remember I hate *copy-pasting*