Example of Codes from the enonce

Objectives

- Create processes with the `fork()` function.
- Handle those processes, especially avoid zombies.
- Handle the signals sent to the shell.

Features

User can

- Interact with system
- Execute commands

Shell must

- Create multiple processes (`fork`)
- Handle those processes (avoid zombies)
- Be able to use builtin commands (`cd <dirname>` and `exit <exit-code>`)
- Be able to execute jobs, i.e. system's programs (e.g. `ls` , `pwd` , `ps` ...)

Parsing

Comment lire l'entrée utilisateur ?

- Split the input string to an array than can be used like `argv` , `argc` (c.f. `main`)
- In our case we will consider that the space and the tab are the only 2 arguments separators.
    - Consult the `strtok` function in the manuel. The `realloc` function can also be useful.

```
 char* token = strtok(userinput, "");

while (token != NULL) {
  printf("%s\n", token);
  token = strtok(NULL, "");
}
```

Jobs

When the users enters a non-builtin command:

- Shell will create another process to execute the program with the `execve` call

```
 include <unistd.h>
int execve(const char *filename, char *const argv[], char *const envp[]);
```

Shell will wait for the child process to end then display its exit code if it's available. (e.g. "Foreground job exited withcode 0"), or if it is not: a simple message (e.g. "Foreground job exited")