2nd Year – Computer Science Major

Synthesys Practical Work
Ensea in the Shell

Sessions 1 & 2 (8 h)

C. BARÈS

Objectives: Develop a tiny shell, that displays exit codes and execution times of launched programs.

General advices:

- *You are strongly encouraged to write* ***one*** *file per question (by copying the file of the previous question) and to use a Makefile (see Moodle) or even better, to use* ***GIT****.*
- *Use relevant comments (no: `i++; //increment of i`);*
- *Similarly, dividing your program into correctly named functions should improve the readability of your code;*
- *Name your constants, do not use "magic" numbers;*
- *Don't use the `printf`, it doesn't mix well with `read` and `write`;*
- *Pour manipuler les chaînes de caractères, utiliser l'entête `string.h`, and always use the functions starting with `strn...`*

Create a micro shell, which you will call enseash, to be used for launching commands and displaying information about their execution.

The following features are required, to be done in this particular order:

1. Display a welcome message, followed by a simple prompt. For example:

```
$ ./enseash
Welcome to ENSEA Tiny Shell.
Pour quitter, tapez 'exit'.
enseash %
```

2. Execution of the entered command and return to the prompt (REPL: read–eval–print loop):

   a) read the command entered by user,
   b) execute this command (simple command for the moment, without argument)),
   c) print the prompt `enseash %` and waits for a new command

```
enseash % fortune
Today is what happened to yesterday.
enseash % date
Sun Dec 13 13:19:40 CET 2020
enseash %
```

3. Management of the shell output with the command ”exit” or with `<ctrl>+d`;

```
    enseash % exit
    Bye bye...
    $
```

4. Display the return code (or signal) of the previous command in the prompt:

```
    enseash % un_programme
    enseash [exit:0] % un_autre_programme
    enseash [sign:9] %
```

5. Measurement of the command execution time using the call `clock_gettime`:

```
    enseash % un_programme
    enseash [exit:0|10ms] % un_autre_programme
    enseash [sign:9|5ms] %
```

6. Execution of a complex command (with arguments);

```
    enseash % hostname -i
    10.10.2.245
    enseash % fortune -s osfortune
    "However, complexity is not always the enemy."
      -- Larry Wall (Open Sources, 1999 O'Reilly and Associates)
    enseash %
```

7. Management of redirections to `stdin` and `stdout` with ’<’ and ’>’;

```
    enseash % ls > filelist.txt
    enseash [exit:0|1ms] % wc -l < filelist.txt
    44
    enseash [exit:0|4ms] %
```

8. Management of pipe redirection with ‘|’:

```
    enseash % ls | wc -l
    44
    enseash [exit:0|5ms]%
```

9. Return to the prompt immediately with ‘&’ (execution of programs in the background):

   a) Define a data structure for background process management,
   b) Use of a non-blocking `wait` for background processes,
   c) Management of information display for background programs
   d) Correction of execution time measurement (call to `wait4`).

```
    enseash % sleep 10 &
    [1] 3656
    enseash [1&] %
    [1]+  Ended: sleep 10 &
    enseash [exit: 0|10s] %
```