B1 – Unix System Programming

minishell1

Bootstrap

{EPITECH.}

minishell1

binary name: my_exec
language: C
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

STEP 1: WE LIED TO YOU

Print the content of env

```
int    main(int argc, char **argv, char **env)
{
    ...
}
```

STEP 2: A SIMPLE EXECUTION

With the step 1 done, add a program that execute /bin/ls using execve.

> man execve

STEP 3: SEGMENTATION (NOT FAULT)

Now we are going to prepare your program for the next step.
Your program has to take one parameter (a program name with its path and arguments) and transform into a char **

```
void    print_arg(char **arg)
{
    while (*arg)
        {
            my_putstr(*arg);
            my_putchar('n');
            arg++;
        }
```

{ EPITECH. }

}

```
▽                              Terminal                          –  +  X
~/B-PSU-101> ./my_exec "/bin/ls -l /dev"
/bin/ls
-l
/dev
```

STEP 4: EXECUTE

Now rewrite your program whose has to take one string as parameter, that contains a program name with its path and arguments.
Your program must execute the program with these arguments and display as following:

```
┌─────────────────────────────────────────────────────────────────────────┐
│ ▽                              Terminal                         - + x     │
├─────────────────────────────────────────────────────────────────────────┤
│ ~/B-PSU-101> ./my_exec "/bin/ls -l /dev"                                  │
│ Program name: /bin/ls                                                     │
│ Nb args: 2                                                                │
│ PID: 1346                                                                 │
│ Child PID: 1348                                                           │
│ ... . .                                                                   │
│ ... . .                                                                   │
│ ... . . /* execution of ls -l /dev */                                     │
│ ... . .                                                                   │
│ ... . .                                                                   │
│ Program terminated.                                                       │
│ Status: OK                                                                │
└─────────────────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────────────────┐
│ ▽                              Terminal                         - + x     │
├─────────────────────────────────────────────────────────────────────────┤
│ ~/B-PSU-101> ./my_exec "./my_segfault"                                    │
│ Program name: ./my_segfault                                               │
│ Nb args: 0                                                                │
│ PID: 1513                                                                 │
│ Child PID: 1514                                                           │
│ ... . .                                                                   │
│ ... . .                                                                   │
│ ... . . /* execution of ./my_segfault */                                  │
│ ... . .                                                                   │
│ ... . .                                                                   │
│ Program terminated.                                                       │
│ Status: Segmentation fault                                                │
└─────────────────────────────────────────────────────────────────────────┘
```

Status: is the return value of execve

Authorized Functions

- malloc, free, exit, opendir, readdir, closedir, getcwd, chdir
- fork, stat, lstat, fstat, getcwd, open, close
- read, write, execve, access, isatty, wait, waitpid
- wait3, wait4, signal, kill, getpid, strerror, perror, strsignal