BOOTSTRAP - MY_TOP

LET'S TAKE A LOOK AT UNIX PROCESSES !

BOOTSTRAP - MY_TOP

language: C
compilation: gcc *.c
Forbidden functions: system, exec*, popen, getloadavg, getrusage, getrlimit, getutent, setutent or any other function that retrieves process or system information for you.

- ✓ The totality of your source files, except all useless files (binary, temp files, objfiles,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ 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).

Reminder

The my_top project is about retrieving system and process information and statistics within **procfs** and display those within a terminal thanks to the **ncurses** library.

This bootstrap will cover:
- getting informations from /proc
- ncurses window
- writing within the ncurses window
- getting keyboard events
- writing and updating those informations

For the first part of this bootstrap, you will be using the ncurses library.
man pages ncurses(3), initscr(3), waddchr(3), waddstr(3), wprintf(3) are your friends.

You will find that ncurses(3) contains a large list of other man pages for plenty of various functions

After that, we will start exploring procfs, retrieving some data, and using it within our ncurses application.
man page procfs(3) will help you.

Procfs

Step 1

Let's start by exploring **procfs** ! Write a function that retrieves the load average.

This function should be prototype like this:

```
int my_getloadavg(double loadavg[], int nelem);
```

It should fill the `loadavg` array with `nelem` elements found in the load average *procfs* file, and returns -1 if any error occured, 0 otherwise.

This **procfs** man page is pretty heavy... And, spoiler alert, so are the ncurses man pages ! It might be a good idea to learn how to use man pages properly... How about searching through them ? Ask your best friend Google !

If you look at the forbidden functions, you may notice that there is a real `getloadavg` function. Maybe its man page is interesting as well ?

Ncurses

Step 2

Now, let's create a program that initializes an **ncurses** screen.

In the center of this screen, display the load average information retrieved thanks to your `my_getloadavg` function.
You should also find a way not to exit the program. Otherwise your program will close, and you won't see what you wrote. You could use a loop, or simply wait for a fixed amount of time.

Take some time to read through the **ncurses** and **initscr** man pages. The **Initialization** section of the **ncurses** man page should be particularly helpful.

Once again, large man pages. But I'm sure than if you *search* through it, you'll find valuable information about how to *init* a screen and how to *print* stuff.

Hey, if you're not careful, you may "break" your terminal, and not be able to type anything. Don't panic ! You can actually type, it just doesn't show. Try typing `reset` and press `Enter` and things should go back to normal :D
To avoid this, maybe you should close your ncurses application cleanly.

Step 3

Now, that's all nice and well, but your application doesn't close very... elegantly.
Maybe we should find a way to **get characters** from the user, and leave our program when the user types `Q` ?

Come on, ask Google, I'm sure you'll find a way.

Step 4

Ok this is cool, but for some reason, it seems our application only refreshes when pressing a key...
Maybe we can find a way to refresh at regular intervals ?

v 0.0.1

{EPITECH}