{EPITECH}
LEARN DIFFERENT*

MY_TOP

LET'S TAKE A LOOK AT UNIX PROCESSES !

MY_TOP

binary name: my_top
language: C
compilation: via Makefile, including re, clean and fclean rules
Forbidden functions: system, exec*, popen, getloadavg, getrusage, getrlimit, 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).

Lore

> TODO: LORE. Wink wink @Joffrey

The project

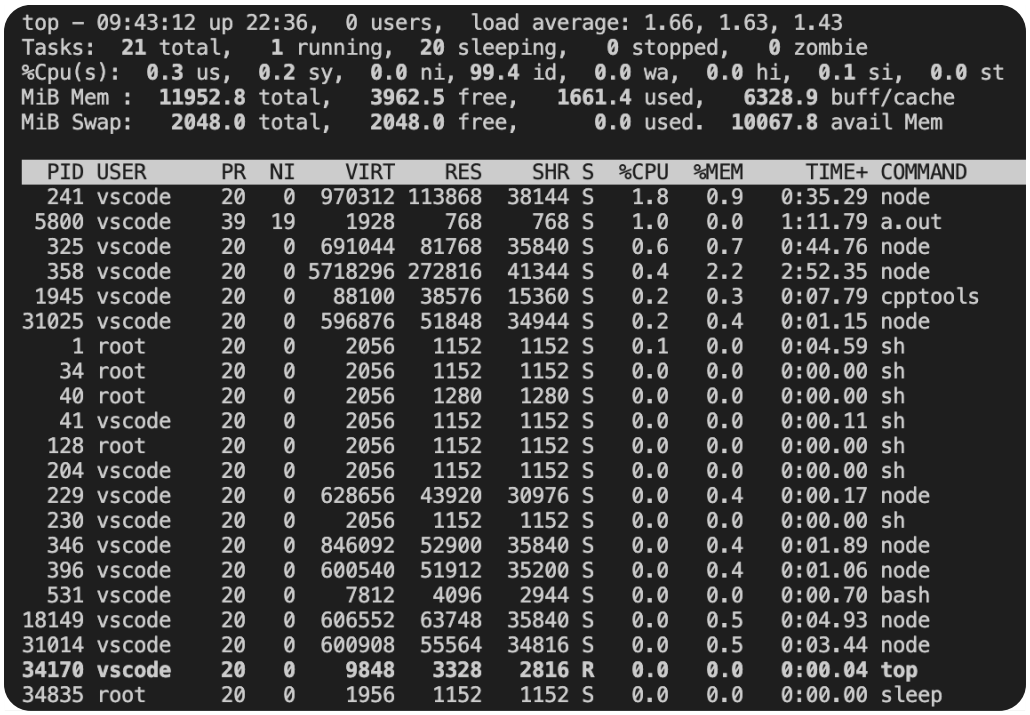
Objectives

You must recode the **top** command.

It is a tool to monitor your system and processes.

Your project will need to use the Terminal User Interface library **ncurses**.

Here is an example of what the `top` command output looks like:



Parameters

You must handle the following options, **in any order**:

- ✓ `-U <username>`: allows to filter the processes shown by username.
- ✓ `-d <seconds[.cents]>`: modifies the delay between refreshes (in seconds).
- ✓ `-n <frames>`: Defines how many frames must show before the program exits (default: unlimited).

This should show processes owned by user `clery` with only one frame.

```
Terminal
~/B-PSU-100> ./my_top -U clery -n 1
```

This should just show processes owned by user `clery`, until the program is manually stopped.

```
Terminal
~/B-PSU-100> ./my_top -U clery
```

This should show processes during 10 refreshes, each refresh at 1.5 second interval.

```
Terminal
~/B-PSU-100> ./my_top -n 10 -d 1.5
```

Features

Your program must be able to retrieve system information, and process statistics.

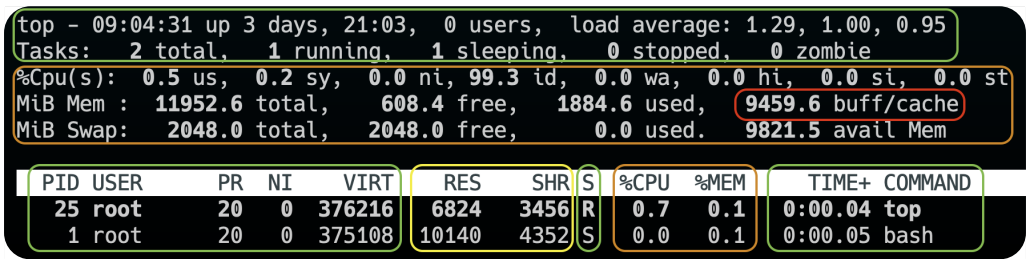
In the upper section of your ncurses window, you must display the system informations. Below that, you must display an array of individual process statistics.

In the upper section you must display:

- ✓ Time of day
- ✓ How long the laptop has been up (powered on)
- ✓ How many users are currently logged in
- ✓ Load average
- ✓ The amount of Tasks total, running, sleeping, stopped or zombie
- ✓ CPU usage (*SHOULD*)
- ✓ Memory usage
- ✓ Swap usage

In the lower section, you must display informations about processes, including:

- ✓ **PID** of the process
- ✓ **USER** owning that process
- ✓ **PR**iority
- ✓ **NI**ce Value
- ✓ **VIRT**ual Memory Size
- ✓ **RES**ident Memory Size
- ✓ **SH**a**R**ed Memory Size
- ✓ Process **S**tatus
- ✓ **CPU** percent usage (*SHOULD*)
- ✓ **MEM**ory percent usage (*SHOULD*)
- ✓ **TIME** since the process has started
- ✓ The **COMMAND** name



The image above describes how tough the different sections are.

This can be due to multiple factors : the research needed to find out how to calculate the results, how to handle data to make it change over time, or any other reason that would make something difficult.

The only red box is because this specific value is tricky, and might require some research.

You should probably start with the easiest columns. CPU and MEM usage are not easy ones !

Formatting exactly like `top` is not mandatory.

You should also implement a number of commands, including the following:

- ✓ Typing **E** should cycle through different units of memory for processes
  Shift+E should cycle through different units of memory for the system section
  - KiB
  - MiB
  - GiB
  - TiB
  - PiB
  - EiB (only for the system section)
- ✓ Using **up and down arrows** must allow you to scroll through your process list
- ✓ Typing **K** should open a prompt to send a signal to a process
  - By default, it should offer to send the signal to the **highest process in your list**
  - By default, it should offer to send the `SIGTERM` signal
  - You are not required to provide line edition

man top
man procfs
top -> hit H on your keyboard to open the documentation

Scrolling down or up changes the **highest** process in your list. So the default PID to send a signal to with the K command **should change** !

Sending a signal should not be your highest priority. You probably want to implement that later on.

But, really, I mean it. Read the manuals for **top** and **procfs**. I swear they're useful.

Bonus

There's so many possible bonuses it would be longer to enumerate them than writing the rest of the subject… But here are a few ideas:

- ✓ Sort rows by column value
- ✓ Using **<** and **>** could change which column the array is sorted by (default PID)
- ✓ Typing **Shift+R** could change the sorting direction
  - – You could sort from highest to lowest value by default
- ✓ More columns ? **PPID, GROUP, UID, GID**. There's so many, just read the man you'll find some easy ones
- ✓ Scrolling horizontally ? Try it out in `top`, it's probably not that hard.
- ✓ Did you try the **Z** command ? Some colors would make that window look better.
- ✓ Forest mode ? **Shift+V**
- ✓ Threads ? **Shift+H**
- ✓ Show the entire command line ? **C**
- ✓ …
- ✓ **htop** ?…

Unit tests

TODO

v 0.0.1