



# NAVY'S BOOTSTRAP

THE UNIX'S BAT SIGNAL



# NAVY'S BOOTSTRAP

## Preliminaries



**binary name:** process\_info, kill\_it, who\_sig\_me, signal\_me  
**language:** C  
**groupe size:** 1  
**compilation:** via Makefile, including re, clean and fclean rules



- ✓ 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).

## Projects

This Bootstrap is an introduction to the way signals works on an Unix environment.

The Unix kernel “informs” processes through signals in order to transmit potential problems (SIG-SEGV for a segmentation error for instance).

Each signal is assigned a default behavior.

Read the related man pages in order to understand each signal's procces' behavior.#br

Numerous system calls exist to handle signals.

We will purposely not cover a large part of them, so that you can have fun researching them yourself.

## Your bootstrap

### process info

---

Write a program named **process\_info** that displays the following process information: process ID, parent process ID and process group ID.#br

For instance:

```
Terminal
$> ./process_info
PID: 18975
PPID: 18954
PGID: 18973
```



man getpid

### kill it

---

Write a program named **kill\_it** that sends the SIGQUIT signal to the process whose PID is passed as parameter.

For instance:

```
Terminal
$> ./kill_it 4754
```



kill man page. In order to test it, use the ps program to obtain the PID of a program (firefox, xeyes etc.)

### who sig me ?

---

Write a program named **who\_sig\_me** that, for each received signal, displays its name and the PID of the emitter process.

The program takes the list of signals to be rerouted as parameter. If a signal's rerouting fails, an error message is displayed.

For instance:

```
Terminal
$> ./who_sig_me 10 12 > stdout.log &
[1] 1585
$> kill -USR1 1585
$> kill -USR2 1585
$> kill -USR1 1585
$> kill -QUIT 1585
$> cat -e stdout.log
Signal User defined signal 1 received from 1586
Signal User defined signal 2 received from 1587
Signal User defined signal 1 received from 1588
```

```
Terminal
$> ./who_sig_me 12 9 > stdout.log &
[2] 1590
$> kill -USR2 1590
$> kill -KILL 1590
$> cat -e stdout.log
Unable to handle Killed signal
Signal User defined signal 2 received from 1591
```



sigaction and strsignal man pages. Scan the system's .h in order to understand the number that is associated with each signal.

## signal me

Write a program named **signal\_me** that counts the number of times it receives the SIGUSR1 and SIGUSR2 signals.

This program must display a summary when it receives the SIGQUIT signal, before exiting.

```
Terminal
./signal_me > stdout.log &
[1] 12985
$> kill -USR1 12985
$> kill -USR1 12985
$> kill -USR2 12985
$> kill -USR2 12985
```

```
$> kill -USR1 12985  
$> kill -QUIT 12985  
$> cat -e stdout.log  
SIGUSR1: 3  
SIGUSR2: 2
```

