Lab 6.1

To extract the content of the **xv6.tgz** tar-zipped file

```
tar vxfz xv6.tgz
```

```
make clean
```

Compile xv6 with **make qemu** this command will also run the **xv6** on **qemu**.

Try some commands (ex. **ls**).

```
make qemu-gdb
```

copy the last line of the screen, something like

```
qemu -serial mon:stdio -hdb fs.img xv6.img -smp 2 -m 512  -S -gdb tcp::26000
```

on a script file **qemu.sh**

Then, run

```
qemu -serial mon:stdio -hdb fs.img xv6.img -smp 2 -m 512
```

and using **cat** and redirection, create a file **myname.txt** including a single string: **<your name>**.

check that a **.gdbinit** file exist that refers to the same tcp port (**26000**)

run **./qemu.sh** on a window

run **ddd&** on another window

Write a report that lists and comments the sequence of system calls

that are performed after issuing the command

```
cat myname.txt | wc
```

Lab 6.2

Since the xv6 is a kernel without semaphores, take inspiration from the **file system** (**`file.c`**) **and pipe** (**`pipe.c`**) implementation as sources for adding the semaphore system calls:

`int sem_alloc()`

`void sem_init(int sem, int count)`

`void sem_destroy(int sem)`

`void sem_wait(int sem)`

`void sem_post(int sem)`

Hints: modify the files

`param.h`

`user.h`

`usys.S`

`syscall.h`

`syscall.c`

`file.c` to add the **`semaphore`** structure definition, and your **`sem`**_ functions

`sysfile.c` to add the **`sys_sem`**_ functions in analogy with the other **`sys_...`** functions

`main.c` to call **`semaphore_init()`**

`Makefile` to add **`_st`**

Use the main file **`st.c`** to test your semaphores system calls.