Lab 5.1

`Go to the directory xv6`

`make clean`

Compile xv6 with **`make qemu`** it will also run the **`xv6`** on **`qemu`**.

Try some commands (ex. **`ls, cat, …`**).

Exit **`qemu`**

`make qemu-gdb`

Check if the script file **`qemu.sh`** does not exist, in this case, 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 **`qemu.sh`**

Then, run **`qemu`** without suspending it, using

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

Using **`cat`** and redirection, create a file **`test.txt`** including the string**:**

`System and Device Programming`.

Exit **`qemu`**

Notice that if you run again **`qemu`**, the file created is stored in the filesystem (try **`ls`**).

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

```
wc < myname.txt | grep 1
```

Lab 6.2

Introduce the semaphores on xv6 kernel.

Since the xv6 is a kernel without threads, see the file system and pipe implementation as source for developing your semaphores.

Add to the xv6 Makefile the main file **`st.c`**, which tests the semaphores system calls

Modify the files

`Makefile`

`param.h`

`user.h`

`usys.S`

`syscall.h`

`syscall.c`

`file.c`

`sysfile.c`

`main.c`

to add the 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)
```