CS4375-13948 Fall 2023
Edmundo Pariente
eipariente@miners.utep.edu

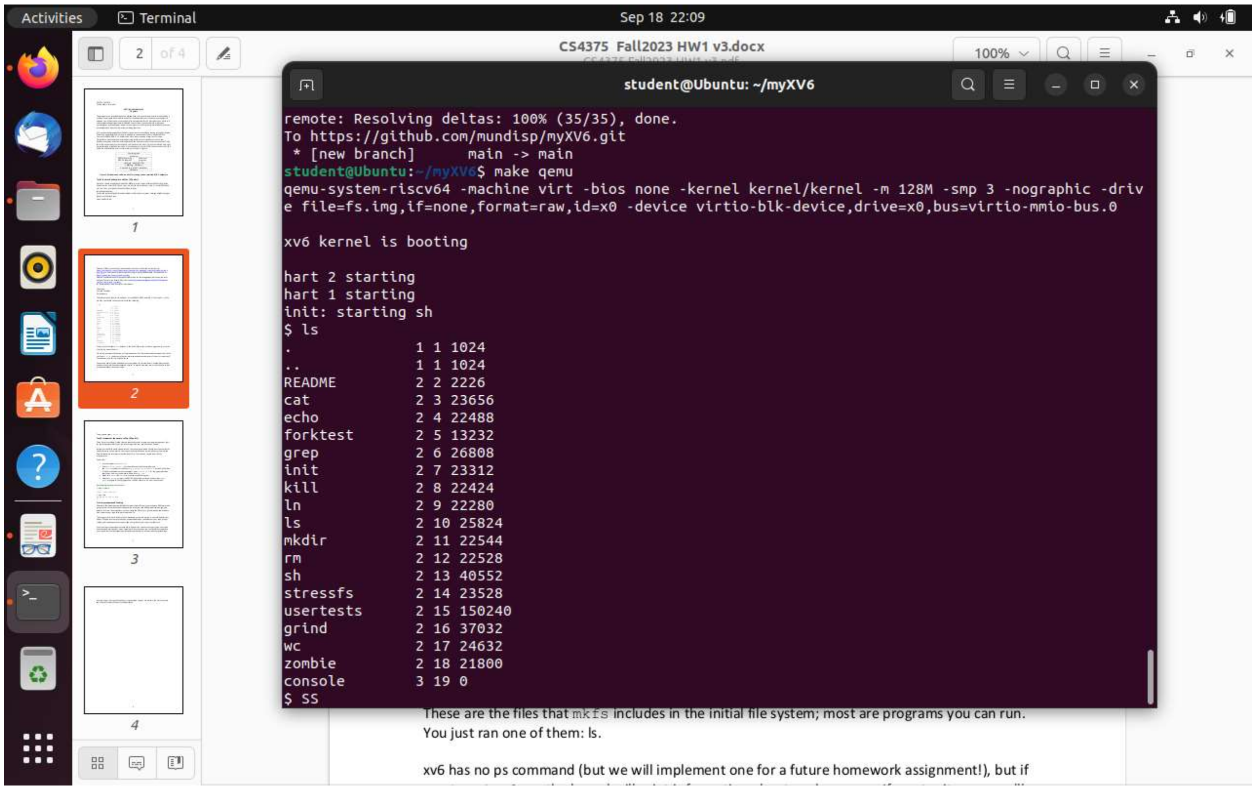
Homework Report
Submitted on **9/22/2023**

HW 1: Introduction to xv6

Task 1. Boot xv6 and explore utilities.

The linux setup that I'm using is the Ubuntu operating system running on the virtual box software (vmm).

Booting xv6 on RISC-V & running 'ls' command:
Booting xv6 was easy overall, the only issue I had is that I didn't have supervisor privileges when running the 'sudo' command, I had to access some of the OS settings with special commands I found online to be able to include my username into the sudo directory, then I was able to run all the instructions smoothly.

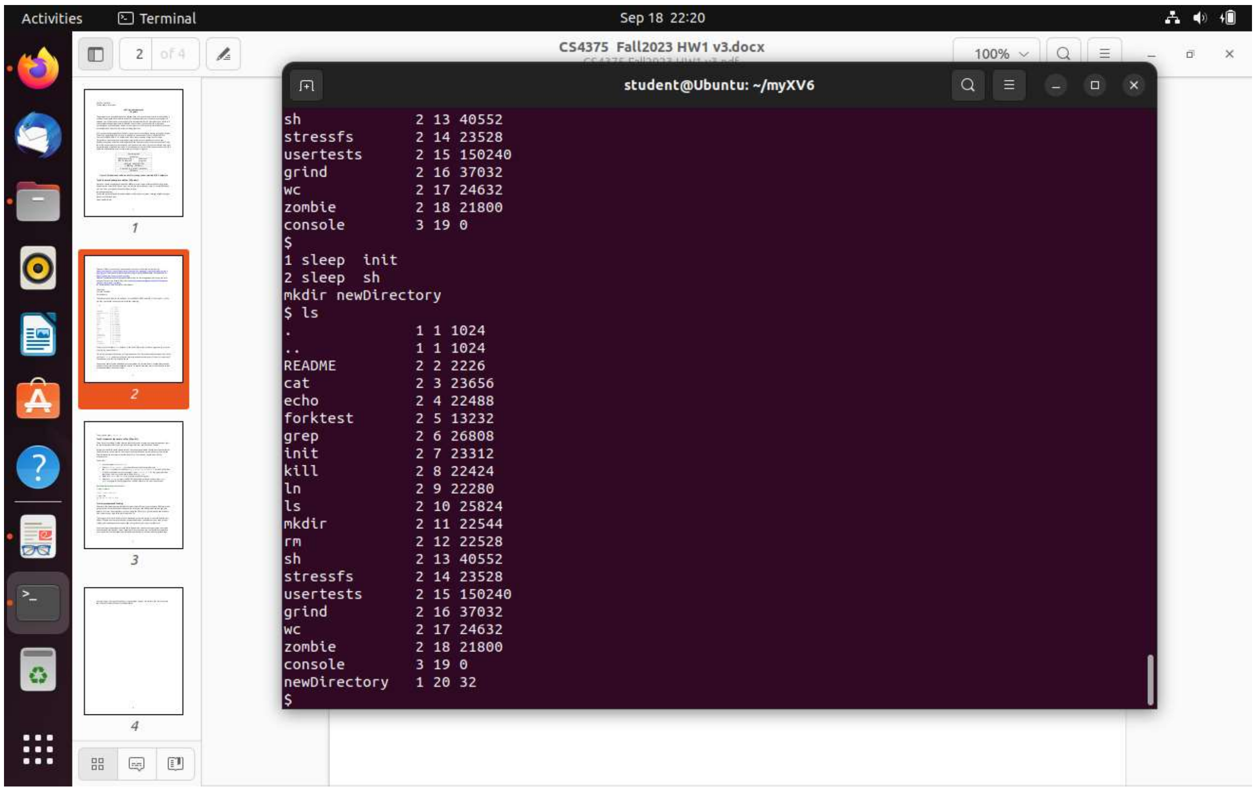


```
remote: Resolving deltas: 100% (35/35), done.
To https://github.com/mundisp/myXV6.git
 * [new branch]      main -> main
student@Ubuntu:~/myXV6$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -driv
e file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ ls
.              1 1 1024
..             1 1 1024
README         2 2 2226
cat            2 3 23656
echo           2 4 22488
forktest       2 5 13232
grep           2 6 26808
init           2 7 23312
kill           2 8 22424
ln             2 9 22280
ls             2 10 25824
mkdir          2 11 22544
rm             2 12 22528
sh             2 13 40552
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
$ SS
```

These are the files that mkfs includes in the initial file system; most are programs you can run. You just ran one of them: ls.

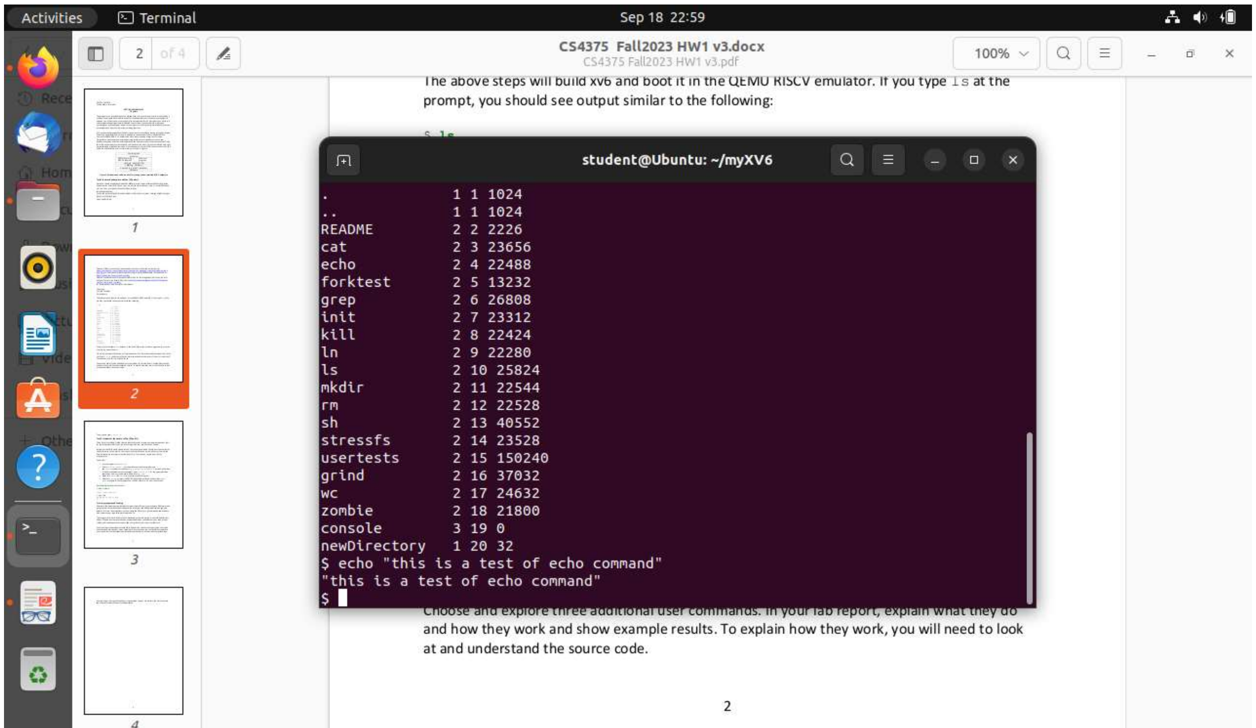
xv6 has no ps command (but we will implement one for a future homework assignment!), but if

Running "mkdir" command:
The mkdir command creates a new empty directory in the location where you type it, you just need to type the command and the name of the folder separated by a space.



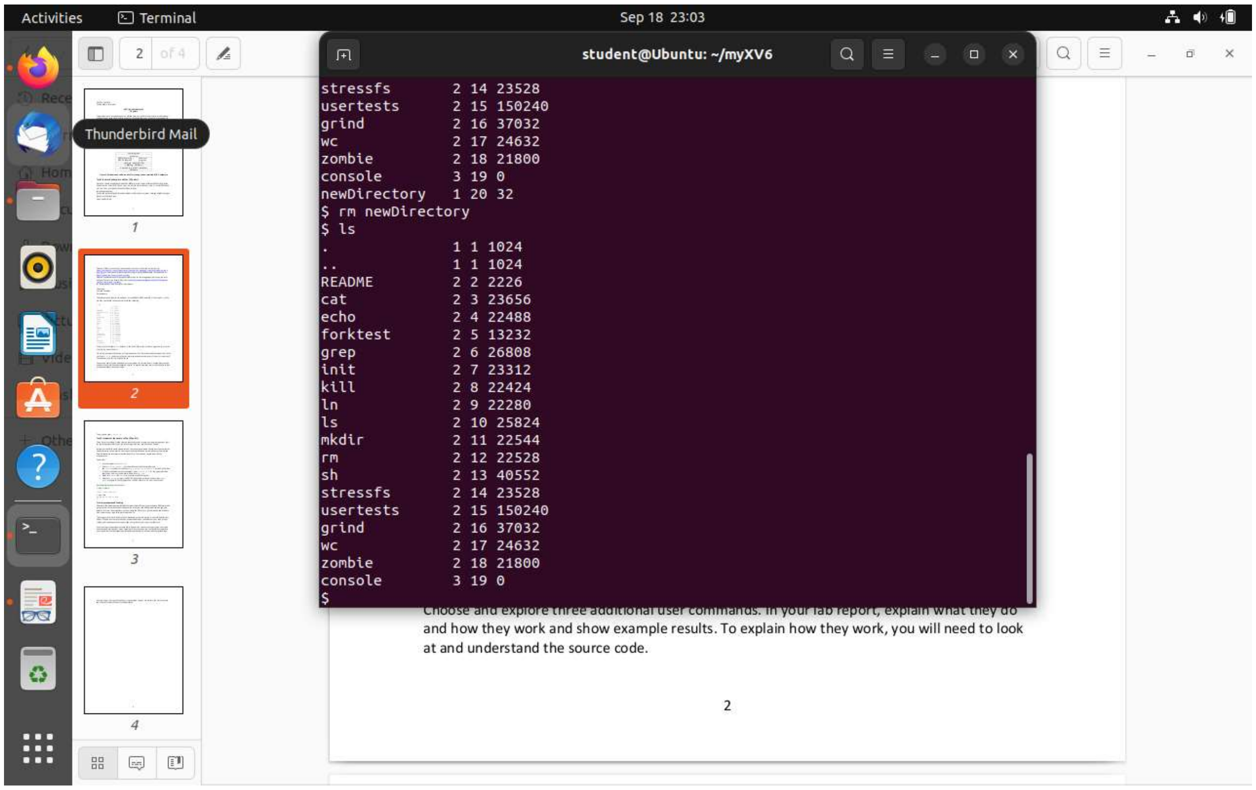
```
sh             2 13 40552
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
$
1 sleep  init
2 sleep  sh
mkdir newDirectory
$ ls
.              1 1 1024
..             1 1 1024
README         2 2 2226
cat            2 3 23656
echo           2 4 22488
forktest       2 5 13232
grep           2 6 26808
init           2 7 23312
kill           2 8 22424
ln             2 9 22280
ls             2 10 25824
mkdir          2 11 22544
rm             2 12 22528
sh             2 13 40552
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
newDirectory   1 20 32
$
```

Running "echo" command:
The echo command outputs the arguments provided back to the terminal; you simply type echo followed by the string of words inside the double quotes.



```
.              1 1 1024
..             1 1 1024
README         2 2 2226
cat            2 3 23656
echo           2 4 22488
forktest       2 5 13232
grep           2 6 26808
init           2 7 23312
kill           2 8 22424
ln             2 9 22280
ls             2 10 25824
mkdir          2 11 22544
rm             2 12 22528
sh             2 13 40552
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
newDirectory   1 20 32
$ echo "this is a test of echo command"
"this is a test of echo command"
$
```

Running "rm" command:
The rm command lets the user remove a specified directory, just type rm followed by the name of the folder that you want to delete.



```
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
newDirectory   1 20 32
$ rm newDirectory
$ ls
.              1 1 1024
..             1 1 1024
README         2 2 2226
cat            2 3 23656
echo           2 4 22488
forktest       2 5 13232
grep           2 6 26808
init           2 7 23312
kill           2 8 22424
ln             2 9 22280
ls             2 10 25824
mkdir          2 11 22544
rm             2 12 22528
sh             2 13 40552
stressfs       2 14 23528
usertests      2 15 150240
grind          2 16 37032
wc             2 17 24632
zombie         2 18 21800
console        3 19 0
$
```

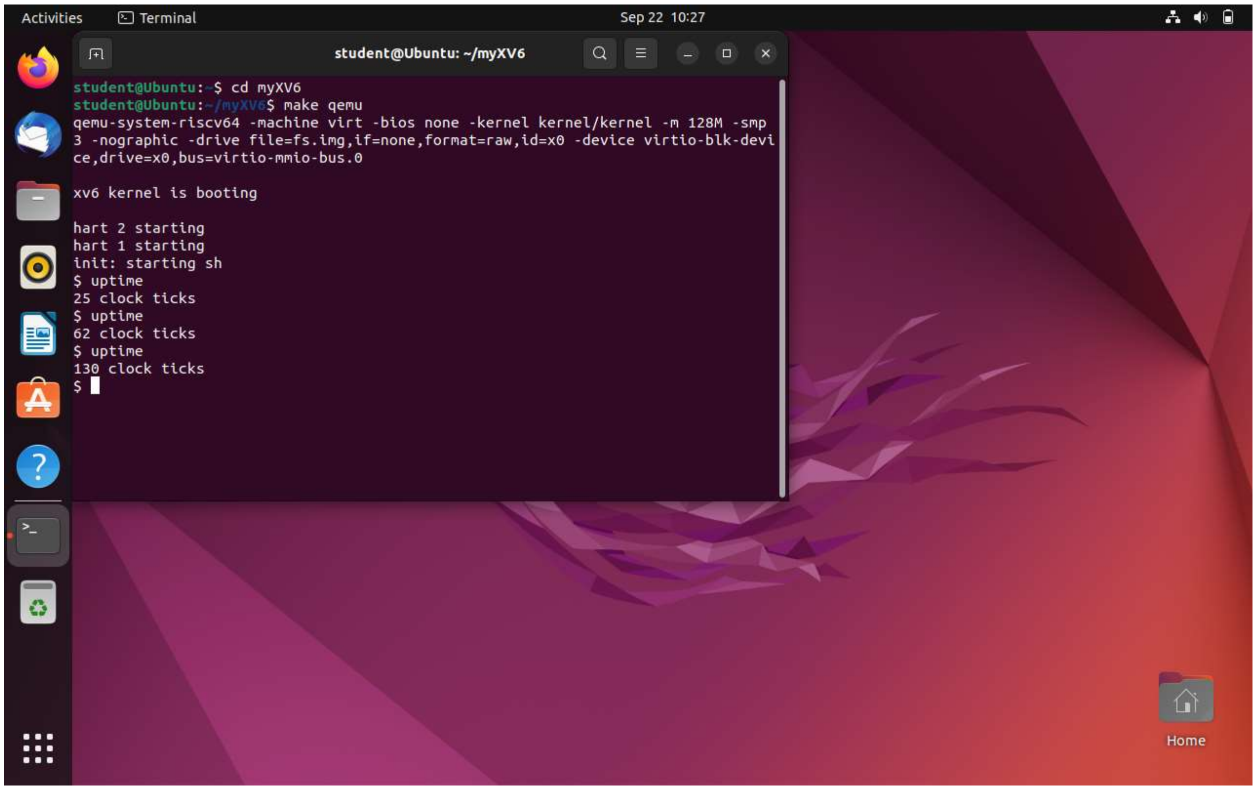
Choose and explore three additional user commands. In your lab report, explain what they do and how they work and show example results. To explain how they work, you will need to look at and understand the source code.

Task 2. Implement the uptime utility.

I learned how to implement an already defined function and turn it into a command. Since the uptime function was already defined in user.h, I had to include that header file so that the code would recognize the calling. Also, I needed to include the path to the types.h file in the kernel directory, this is the header file that defines the data types and sizes of the integer that the specified uptime function uses, otherwise it would give errors.

Then I included the command name in the Makefile so that it would be able to compile it.

Lastly, I just provided the behavior of uptime function in uptime.c, printing the value returned by the uptime function call every time the command is invoked.



```
student@Ubuntu:~$ cd myXV6
student@Ubuntu:~/myXV6$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ uptime
25 clock ticks
$ uptime
62 clock ticks
$ uptime
130 clock ticks
$ 
```