

CS4375-13948 Fall 2023 Homework Report

Ashley Magallanes /Submitted on Sep 16, 2023

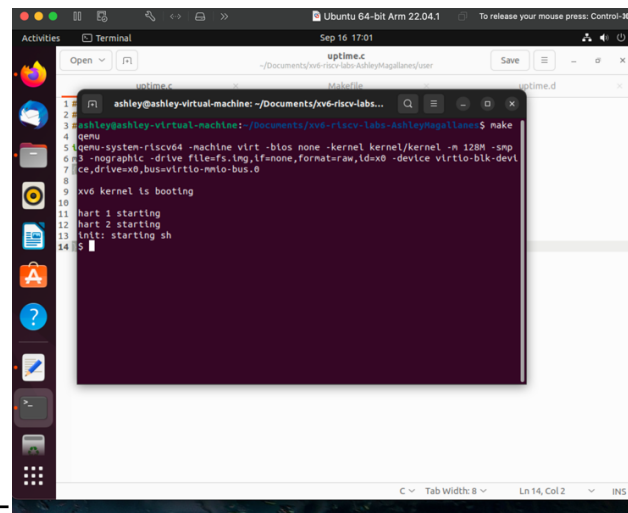
avmagallanes@utep.miners.edu

HW 1: Introduction to xv6

Task 1. Boot xv6 and explore utilities

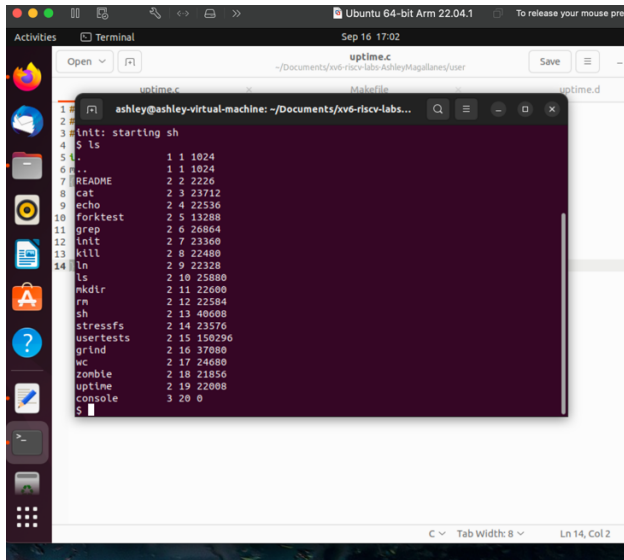
-In order to boot xv6 I obtained Ubuntu server operating system I had to download a virtual machine along with ubuntu itself. I did this through VMware Fusion virtual machine. This allowed me to obtain Ubuntu 64-bit 22.04.1 on my Apple M2 MacBook.

-After duplicating Dr.Moore's repository and setting up my git terminal in Ubuntu, I used the command "make qemu" in order to boot the system. The following image showcases the xv6 kernel booting system.

A screenshot of a terminal window on a Ubuntu 64-bit Arm 22.04.1 system. The terminal shows the execution of the 'make qemu' command, which boots the xv6 kernel. The output displays the boot progress, including the initialization of the hart and the starting of the shell. The terminal window is titled 'ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...' and shows the command history for the 'make qemu' command.

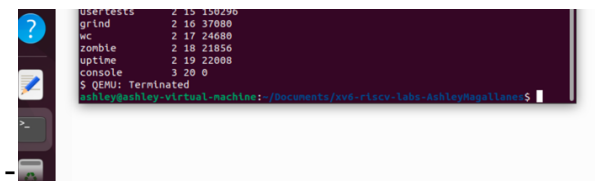
```
ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...  
1 make qemu  
2  
3 ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...$ make qemu  
4  
5 qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp  
6 13 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-dev  
7 ce,drive=x0,bus=virtio-mmio-bus.0  
8  
9 xv6 kernel is booting  
10  
11 hart 1 starting  
12 hart 2 starting  
13 init: starting sh  
14 $
```

-When running the 'ls' process, all the initial files in the system are showcased. This is demonstrated in the following screenshot.



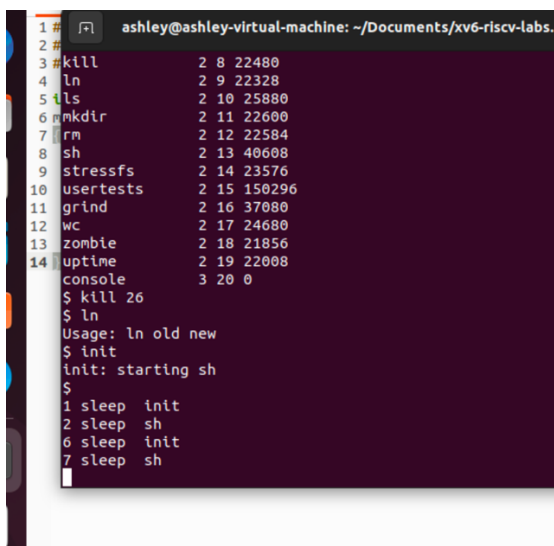
```
ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...
1 #
2 #
3 # init: starting sh
4 $ ls
5 .
6 ..
7 README
8 cat
9 echo
10 forkttest
11 grep
12 init
13 kill
14 ln
15 ls
16 mkdir
17 rm
18 sh
19 stressfs
20 usertests
21 grind
22 wc
23 zombie
24 uptime
25 console
```

-I tried the commands specified on the assignment document, including the command to terminate qemu.



```
usertests 2 15 150296
grind 2 16 37080
wc 2 17 24680
zombie 2 18 21856
uptime 2 19 22008
console 3 20 0
$ QEMU: Terminated
ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs AshleyMagallane $
```

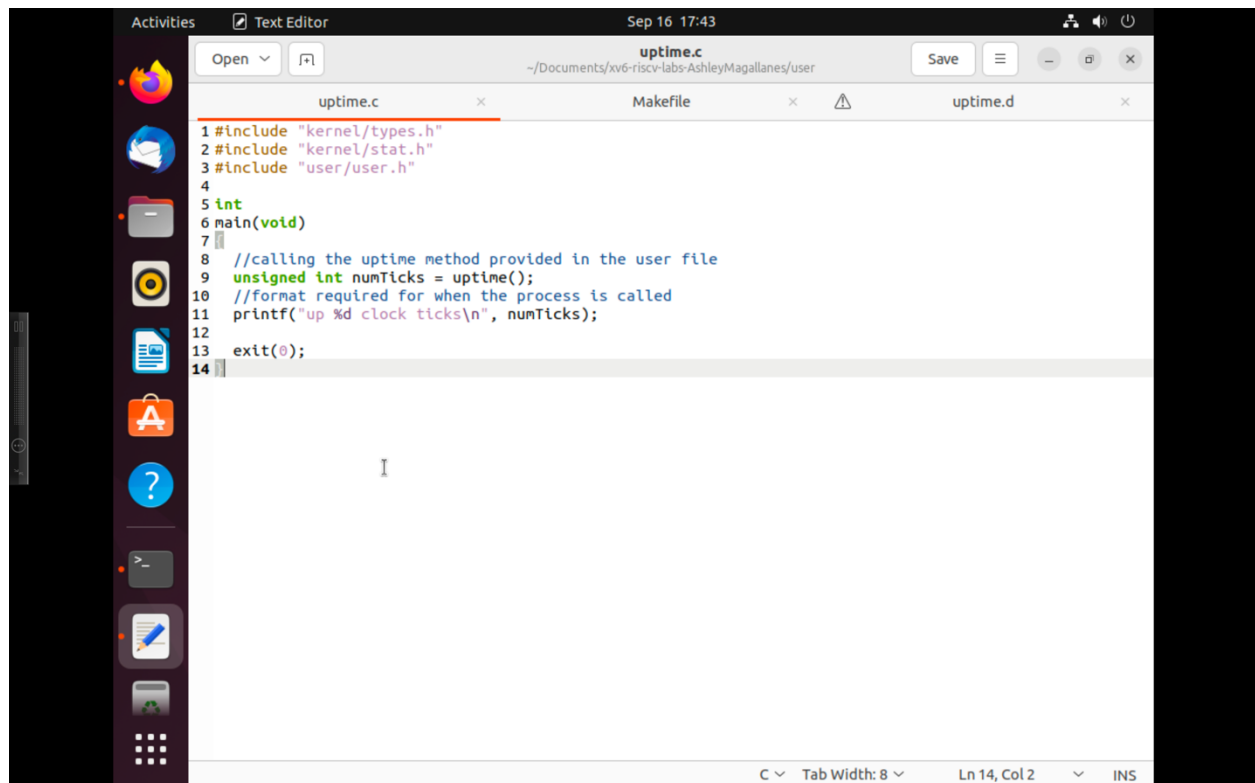
-In addition to the 'ls' command I tried the commands: kill, ln, and init. The 'kill' command just manages to terminate the program. The 'ln' command showcases "Usage: ln old new" and lastly the "init" command starts the "sh" process. Then I also tried the ctrl-p command which produced the sleep schedule for all the processes I did before.



```
ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...
1 #
2 #
3 # kill
4 # ln
5 # ls
6 # mkdir
7 # rm
8 # sh
9 # stressfs
10 # usertests
11 # grind
12 # wc
13 # zombie
14 # uptime
15 # console
$ kill 26
$ ln
Usage: ln old new
$ init
init: starting sh
$
1 sleep init
2 sleep sh
6 sleep init
7 sleep sh
```

-Other than downloading the virtual machine in my M2 MacBook and then setting up the repository, I had a hard time understanding how qemu worked. Since I had never seen it before I was confused. However, after trial-and-error experience I managed to understand it better. Another thing that I had a hard time doing was killing some processes that were not letting my git be installed on the Ubuntu terminal.

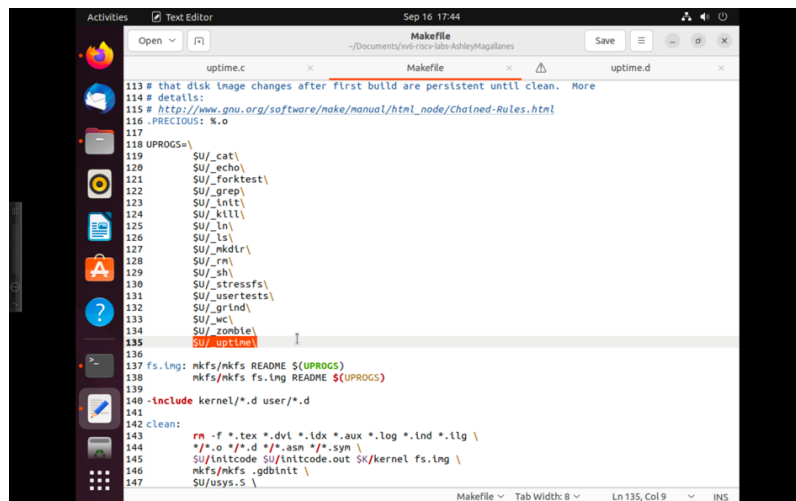
Task 2. Implement the uptime utility

A screenshot of a Linux desktop environment showing a text editor window titled 'uptime.c'. The editor is open to a file named 'uptime.c' located at '~/Documents/xv6-riscv-labs-AshleyMagallanes/user'. The code in the file is as follows:

```
1 #include "kernel/types.h"
2 #include "kernel/stat.h"
3 #include "user/user.h"
4
5 int
6 main(void)
7 {
8     //calling the uptime method provided in the user file
9     unsigned int numTicks = uptime();
10    //format required for when the process is called
11    printf("up %d clock ticks\n", numTicks);
12
13    exit(0);
14 }
```

The editor interface includes a top bar with 'Open', 'Save', and window controls. The left sidebar shows the Ubuntu dock with various application icons. The bottom status bar indicates 'C', 'Tab Width: 8', 'Ln 14, Col 2', and 'INS'.

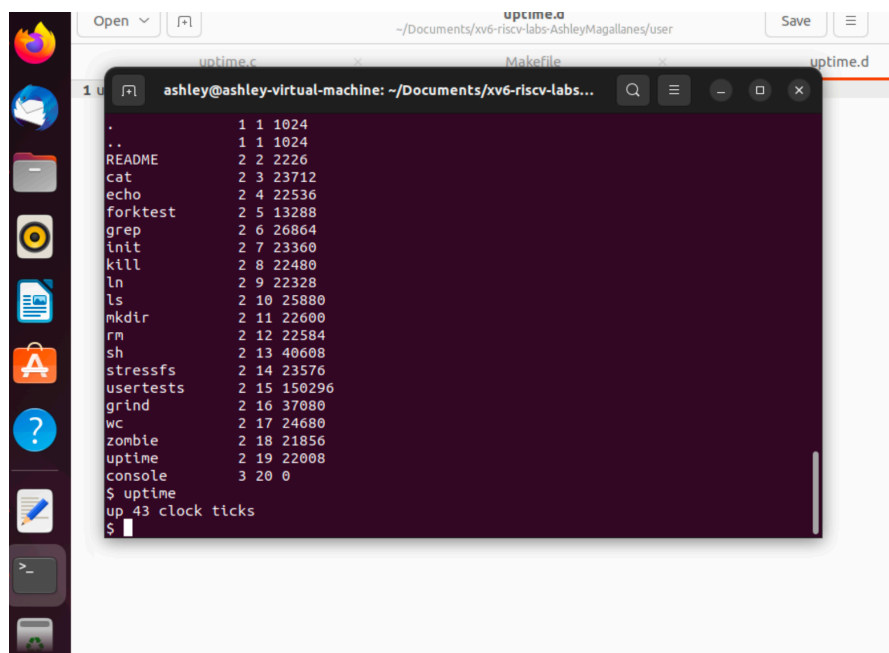
-I Implemented the UNIX utility uptime for xv6 by adding a C file to the user folder on the repository I duplicated. This file calls the method uptime() which was already provided by Dr.Moore, then it prints the outcome in the correct format. This format was not only the one wanted by the assignment instructions, but I tried the “uptime” command on my own MacBook terminal and the format is generally universal. That is something nice to learn.



```
113 # that disk image changes after first build are persistent until clean. More
114 # details:
115 # http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
116 .PRECIOUS: %.o
117
118 UPROGS=\
119     $U/_cat\
120     $U/_echo\
121     $U/_forktest\
122     $U/_grep\
123     $U/_init\
124     $U/_kill\
125     $U/_ln\
126     $U/_ls\
127     $U/_mkdir\
128     $U/_rm\
129     $U/_sh\
130     $U/_stressfs\
131     $U/_usertests\
132     $U/_grind\
133     $U/_wc\
134     $U/_zombie\
135     $U/_uptime\
136
137 fs.lng: mkfs/mkfs README $(UPROGS)
138 mkfs/mkfs fs.lng README $(UPROGS)
139
140 -include kernel/*.d user/*.d
141
142 clean:
143     rm -f *.tex *.dvi *.ids *.aux *.log *.ind *.tlg \
144         */*.o */*.d */*.asm */*.sym \
145         $U/initcode $U/initcode.out $K/kernel fs.lng \
146         mkfs/mkfs .gdbinit \
147         $U/ysys.s \
```

-In order for my new uptime.c file to be included in the booting system I had to add it to the MakeFile already set up. I just added the '\$U/_uptime\' line which allows the program to also consider uptime as a UPROG.

-Lastly, I noticed that all of the other command files like kill and init had a '.d' file corresponding to it. Therefore, I also created one for uptime. Saved everything and run 'qemu' again. Then I tested it worked and the following image illustrates the successful outcome. Resulting in the uptime command showcasing the 'up 43 clock ticks.' Overall, this task was not too confusing and it was easy to follow.



```
ashley@ashley-virtual-machine: ~/Documents/xv6-riscv-labs...
. 1 1 1024
.. 1 1 1024
README 2 2 2226
cat 2 3 23712
echo 2 4 22536
forktest 2 5 13288
grep 2 6 26864
init 2 7 23360
kill 2 8 22480
ln 2 9 22328
ls 2 10 25880
mkdir 2 11 22600
rm 2 12 22584
sh 2 13 40608
stressfs 2 14 23576
usertests 2 15 150296
grind 2 16 37080
wc 2 17 24680
zombie 2 18 21856
uptime 2 19 22008
console 3 20 0
$ uptime
up 43 clock ticks
$
```