

CIS 450/ECE 478: Operating Systems

Project 1: Building xv6



January 24, 2019

Winter 2019

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Srinivas Simhan

Table of Contents

Objective	3
Equipment Used	3
Procedure	3
Source Code	4
Explanation of Code	4
Screenshots	5
Fig.A - xv6 is running	5
Fig. B - "hello.c" working	6
Reflection	6

Objective

- To become familiar with setting up a Virtual Machine
- To become familiar with file management using the Linux VM Terminal

Equipment Used

- Oracle VM VirtualBox
- OSProjects file (given by Professor J. Guo)

Procedure

First I downloaded the VirtualBox and installed it. Then I downloaded the Ubuntu Linux Virtual Machine Image and configured the VirtualBox to allow the “OSVM.ova” file to run. Using the given username and password, I logged into the Linux VM and opened a terminal.

In the terminal, I created a folder called “xv6” and build/compiled source code for a basic “Hello <firstName, lastName>” application. Then I executed the code, where the result came out as “Hello, my name is Srinivas Simhan” (Fig. B).

Source Code

“hello.c”

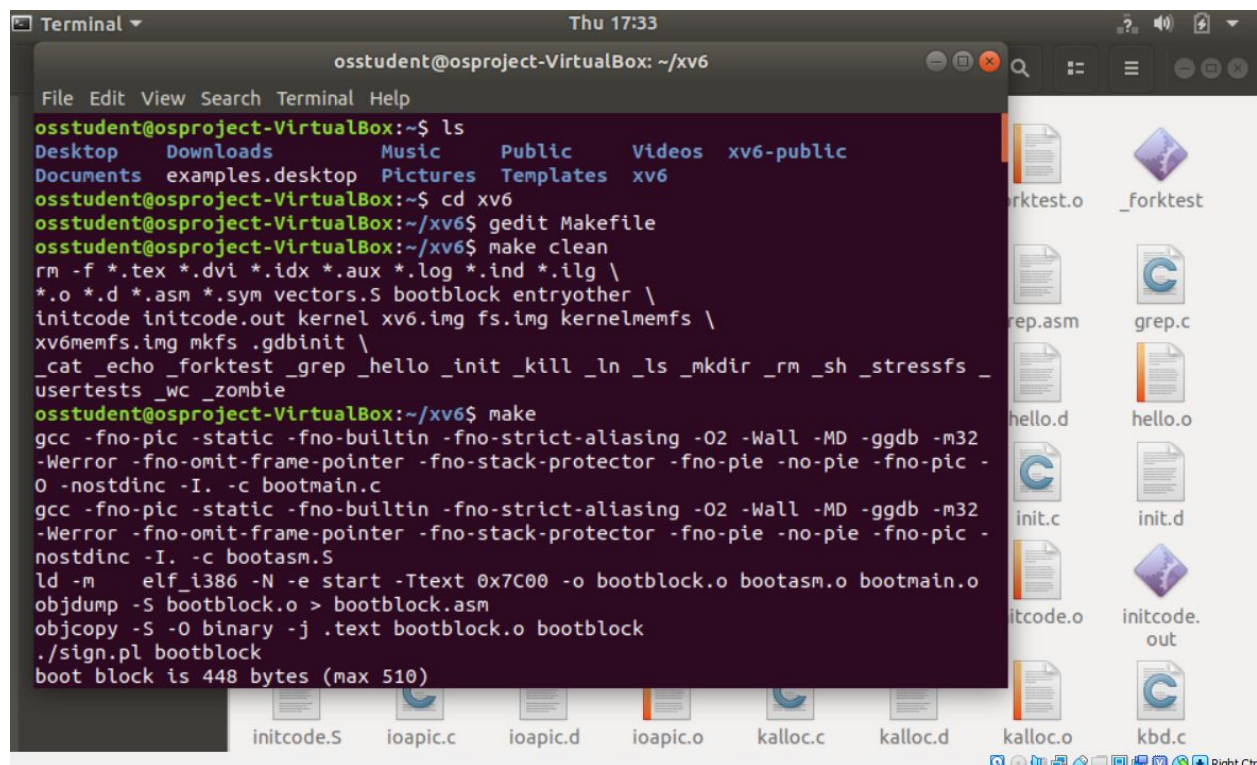
```
#include "types.h"
#include "user.h"
int
main(int argc, char *argv[])
{
    printf (1, "Hello, my name is %s %s\n", argv[1], argv[2]);
    exit();
}
```

Explanation of Code

The “hello.c” code was a simple C programming source code that allowed the user to input two arguments in through the main function, and would output the resulting sentence: “Hello, my name is Srinivas Simhan”.

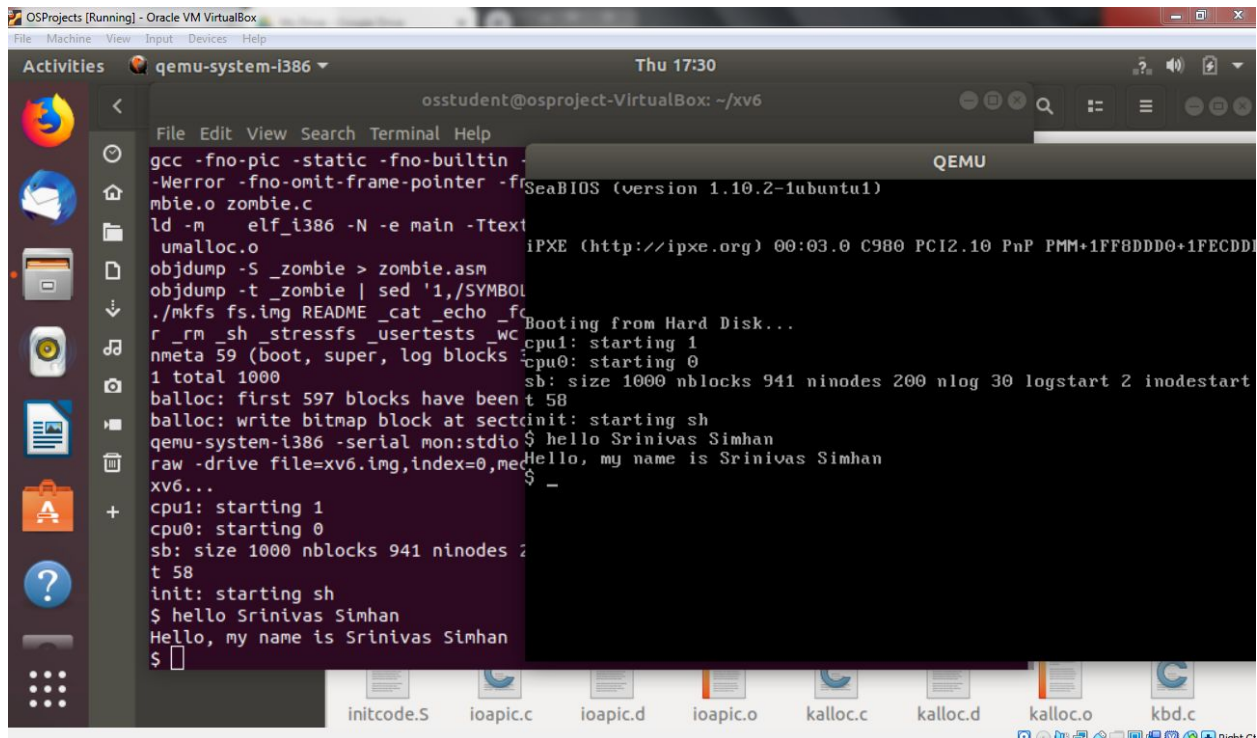
Screenshots

Fig.A - xv6 is running



```
Terminal Thu 17:33
osstudent@osproject-VirtualBox: ~/xv6
File Edit View Search Terminal Help
osstudent@osproject-VirtualBox:~$ ls
Desktop  Downloads  Music  Public  Videos  xv6-public
Documents examples.desktop Pictures Templates xv6
osstudent@osproject-VirtualBox:~$ cd xv6
osstudent@osproject-VirtualBox:~/xv6$ gedit Makefile
osstudent@osproject-VirtualBox:~/xv6$ make clean
rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
*.o *.d *.asm *.sym vectors.S bootblock entryother \
initcode initcode.out kernel xv6.img fs.img kernelmemfs \
xv6memfs.img mkfs .gdbinit \
_cat _echo _forktest _grep _hello _init _kill _ln _ls _mkdir _rm _sh _stressfs _
usertests _wc _zombie
osstudent@osproject-VirtualBox:~/xv6$ make
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32
-Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -
O -nostdinc -I. -c bootmain.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32
-Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -
nostdinc -I. -c bootasm.S
ld -m elf_i386 -N -e start -Ttext 0x7C00 -o bootblock.o bootasm.o bootmain.o
objdump -S bootblock.o > bootblock.asm
objcopy -S -O binary -j .text bootblock.o bootblock
./sign.pl bootblock
boot block is 448 bytes (max 510)
```

Fig. B - “hello.c” working



```

OSProjects [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities qemu-system-i386 Thu 17:30
osstudent@osproject-VirtualBox: ~/xv6
File Edit View Search Terminal Help
gcc -fno-pic -static -fno-builtin -fno-
-Werror -fno-omit-frame-pointer -fno-
mbie.o zombie.c
ld -m elf_i386 -N -e main -Ttext
umalloc.o
objdump -S _zombie > zombie.asm
objdump -t _zombie | sed '1,/SYMBOL
./mkfs fs.img README _cat _echo _fc
r _rm _sh _stressfs _usertests _wc
nmeta 59 (boot, super, log blocks
1 total 1000
ballocc: first 597 blocks have been t
ballocc: write bitmap block at secto
init: starting sh
qemu-system-i386 -serial mon:stdio $
raw -drive file=xv6.img,index=0,med
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 2
t 58
init: starting sh
$ hello Srinivas Simhan
Hello, my name is Srinivas Simhan
$ _
$

```

Reflection

In this project, we learned how to install a Virtual Machine. In this process we got to learn the names of different types of Operating Systems, and that there's more than just Windows and macOS. Something I found really interesting was that it covers a few concepts similar to core Unix concepts, which I haven't really needed to use for class projects since CIS 200, so it was pretty cool to get a refresher on those concepts. This operating system that we used was lightweight, in which the time to compile was very low and it allowed me to debug any issues remotely. This means that if I had any issues or needed to check something out in my notes, I could easily switch over to my Windows 10 screen, and then once I found my answer, I could switch back to the VM. Overall, I enjoyed this project, and I'm looking forward to the future projects we'll have this semester!