Mark Gameng

CS 450 – Duan Yue

## Lab 1

To setup xv6 environment, I used VirtualBox to create a virtual machine running Ubuntu. Once you finish setting up Ubuntu on a VM, you can proceed with the normal instructions to install xv6 on Ubuntu.

The VM screen should look like this

A screenshot of a video game

Description automatically generated

Then, you can go to the terminal using the bar to the right and run the following commands to get xv6.

**sudo apt-get update && sudo apt-get install git nasm build-essential qemu gdb emacs   
git clone** [**https://github.com/mit-pdos/xv6-public.git**](https://github.com/mit-pdos/xv6-public.git) **cd xv6-public  
make  
make qemu-nox**

This launches xv6 and the command line is now running the qemu operating system. To exit out of this press CTRL-A then X.

For me, this command resulted in an error. Saying it couldn’t find a working QEMU executable. Luckily, there was a post on Piazza about fixing this specific problem. The following commands below is how I fixed this problem.

**sudo apt remove qemu  
git clone** [**https://github.com/qemu/qemu.git**](https://github.com/qemu/qemu.git) **cd qemu**  
**sudo apt install -y libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev ninja-build  
./configure --disable-kvm --target-list="i386-softmmu x86\_64-softmmu"   
make  
sudo make install**

After all of this, you can now find qemu-system-i386 on ur system and can now launch xv6 by **make qemu-nox**

**make qemu-gdb**

The command above is to debug xv6 with gdb. To exit out of this press CTRL-A then X. Then in another window go in the same directory and you can enter gdb in the cmd or in emacs.

------------------------------------------------------------------------------------------------------------------------------------------

Below are some pictures of me compiling xv6 and running commands and doing some debugging.

Text

Description automatically generated

Running **make qemu-nox** …

Text

Description automatically generated

Text

Description automatically generated

Running **make qemu-nox-gdb** and debugging

Graphical user interface, text

Description automatically generated

Need to add autoload. So I ran **echo "add-auto-load-safe-path $HOME/xv6-public/.gdbinit" > ~/.gdbinit**

Some debugging using **echo.**

Graphical user interface, text, application

Description automatically generated

By having a breakpoint at main, I can step through **echo.c** and look at what it is doing. Using **s** runs the next line of the program.

Now I am at **printf(…)** as indicated by the black line. The red line is the breakpoint.

Text

Description automatically generated

**s** one more time will go inside printf method and start stepping through the code there.

Running **print** in gdb also allows me to run expressions and print addresses and variables and such.

Text

Description automatically generated

Stepping more and more, you will go to **putc** method, which eventually ends up to **SYSCALL**. This is now kernel debugging.

Text

Description automatically generated

As you can see there is **?? ()**. Now that we are kernel space, we need to change **symbol-file.**

Text

Description automatically generated with medium confidence

Now we know where we are. Stepping through more results in **trap.c** which runs **syscall()**

**Text

Description automatically generated**

Continue stepping along and you can see it progressively prints out “CS450” one character at a time due to running “echo CS450”

Graphical user interface, application

Description automatically generated