Purpose

The purpose of this guide is to help the students getting started with xv6 OS. In the first part of this guide, you will learn how to install xv6 OS on your system. In the second part, you will write a hello world program for the xv6 OS. It is strongly advised that you complete xv6 installation and set up as soon as possible.

What is xv6?

xv6 is an operating system developed at MIT for educational purposes. All the course assignments and project would be based in xv6 and it would help the students understand how the operating system behaves at low level. For more details, please check xv6 [documentation](#)

Part 1: Installation

There are many options for installing xv6 and running and compiling your own programs in it. It is recommended that you follow the method prescribed in the guide. However, depending upon the OS and personal preference, students can explore other options if they want. In this guide, we would be installing the xv6 using Docker. *Some other options are using a VirtualBox and even WSL (windows sub system for linux) for Windows Users.*

Step 1:

Download docker desktop from the official [website](#) and install on your system. It is available for Windows, Linux and Mac (mac users need to select the specific install pertaining to chip). After downloading the setup, please install it on your system. It is a straightforward process and you do not have to use any custom setting options (use the standard setting). After successfully installing, run Docker Desktop

Step 2:

Run this command in PowerShell or bash to pull the ubuntu image with xv6:

docker pull grantbot/xv6

Step 3:

To run the docker image and get going with xv6 run this command:

docker run -it grantbot/xv6

Step 4:

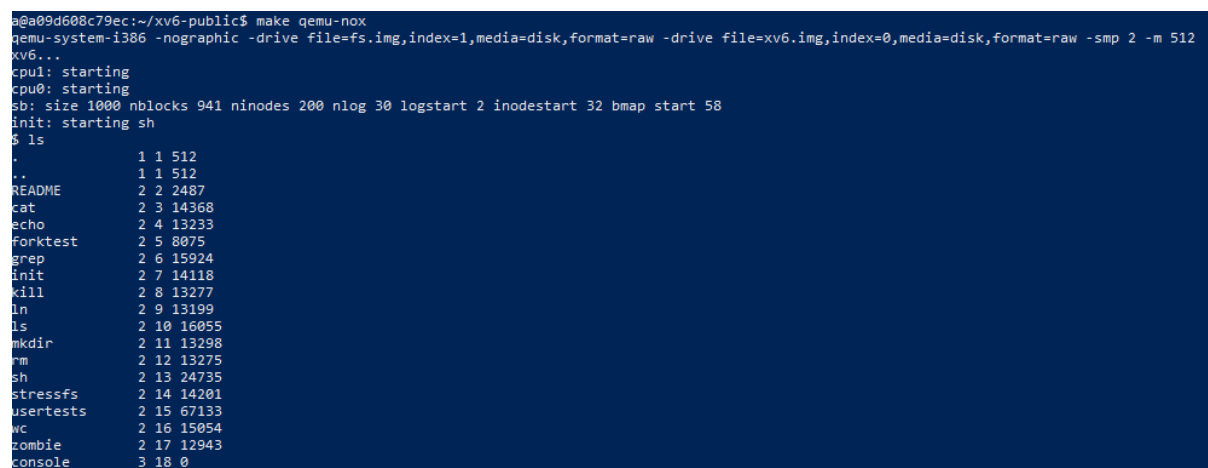Now inside the shell in the ubuntu image run

cd /home/a/xv6-public/

to enter the root folder of the xv6

Step 5:

make (for building the OS)

make qemu-nox (for running the xv6 OS)

You can play around with xv6 for a while. For instance, if you type "ls", it should list you the following output


```
a@a09d608c79ec:~/xv6-public$ make qemu-nox
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2487
cat            2 3 14368
echo           2 4 13233
forktest       2 5 8075
grep           2 6 15924
init           2 7 14118
kill           2 8 13277
ln             2 9 13199
ls             2 10 16055
mkdir          2 11 13298
rm             2 12 13275
sh             2 13 24735
stressfs       2 14 14201
usertests      2 15 67133
wc             2 16 15054
zombie         2 17 12943
console        3 18 0
```

Terminate xv6: pressing control + A at the same time => release => quickly =>press x.

Delete xv6 executable: *make clean*

NOTE: Every time you make change to xv6 code, you need to run "make" (compile xv6) => "make qemu-nox" (launch xv6) => run your program (such as hello) => "make clean" (clean up your executable xv6). Repeat this process when necessary.

Integrating Code Editor

Coding in the Ubuntu linux shell might not be a user-friendly experience. It is suggested that a code editor (Visual Studio Code) should be installed and integrated with xv6

Step 1: Download and install visual studio code from official website

Step 2: Run Visual Studio Code

Step 3: Install the Dev Containers extension (extensions icon can be found on the left pan)

Step 4: From File menu, click Open Folder and choose container (the container of docker already running xv6). If the container is not started, visual studio code would prompt you to start the container and you can press OK to start your container from visual studio code.
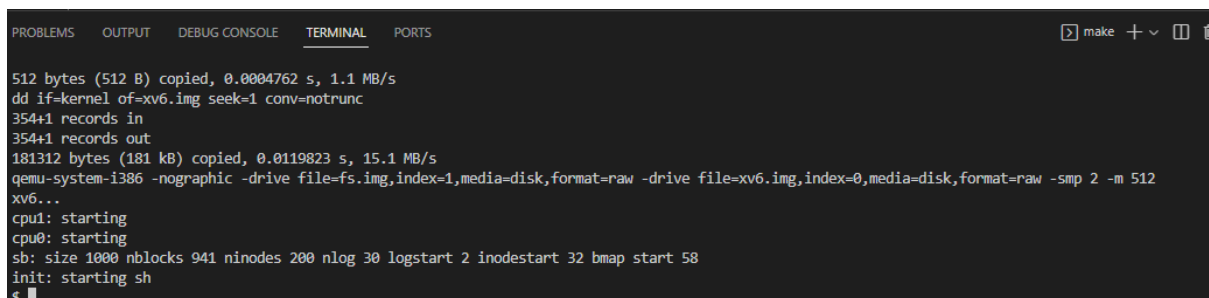
Step 5: Once container is opened, click Terminal->New Terminal. It should take you to the home directory of Linux Ubuntu. Type *cd /home/a/xv6_public*

Step 6: run

make

make qemu-nox

You should be able to access the container from visual studio code.

Terminate xv6: pressing control + A at the same time => release => quickly press x.

Trouble shooting: This link provides a guide for installing container extension with visual studio code and should help in case you run into some problem

Part 2: Running a hello world program in xv6

This is not a graded task but it would help you understand the xv6 OS and assist you in the coming assignments. Write a program for xv6 that, when run, prints "Hello world" to the xv6 console. This can be broken up into a few steps:

a- Create a file in the xv6 directory named hello.c. If you are using the visual studio code, then you can create the file in it using the Visual Studio GUI on the left pan of. If you prefer working in shell, then you can use the vi editor.

b-Put code you need to implement printing "Hello world" into hello.c

c-Edit the Make file (inside the xv6_public directory) and the section UPROGS (which contains a list of programs to be built), and add a line to tell it to build your Hello World program. When you're done that portion of the Make file should look like:

```
UPROGS=\
    _cat\
    _echo\
    _forktest\
    _grep\
    _init\
    _kill\
    _ln\
    _ls\
    _mkdir\
    _rm\
    _sh\
    _stressfs\
    _usertests\
    _wc\
    _zombie\
    _hello\
```

d- Run make to build xv6, including your new program. You can use the same commands described above (*make and make qemu-nox* for building and running until you have compiling code)

e- Run *make qemu-nox* to launch xv6, and then type hello in the QEMU window. You should see "Hello world" be printed out

Of course the second step is where the bulk of the work lies. You will notice that many things are subtly different from the programming environments you've used before; for example, the printf function takes an extra argument that specifies where it should print to. This is because you're writing programs for a new operating system, and it doesn't have to follow the conventions of anything you've used before. To get a feel for how programs look in xv6, and how various APIs should be called, you can look at the source code for other utilities: echo.c, cat.c, wc.c, ls.c.

Hints:

In places where something asks for a file descriptor, you can use either an actual descriptor (i.e., the return value of the open function), or one of the standard I/O descriptors: 0 is "standard input", 1 is "standard output", and 2 is "standard error". Writing to either 1 or 2 will result in something being printed to the screen.

The standard header files used by xv6 programs are types.h (to define some standard data types) and user.h (to declare some common functions). You can look at these les to see what code they contain and what functions they define.

Please let us know as early as possible if you run into any issues in installation and running xv6. It is strongly advised that you complete this task as soon as possible.