Introduction to xv6 – Part I

The xv6 operating system was created at MIT in 2006 to use for teaching Operating System Engineering. It is a modern reimplementation of an early version of Unix, specifically Sixth Edition Unix (or V6) which was originally released in 1975. Xv6 comes with several Linux-like commands, but their features are significantly limited. We will be using the Linux and xv6-riscv operating systems to practice the concepts of this class.

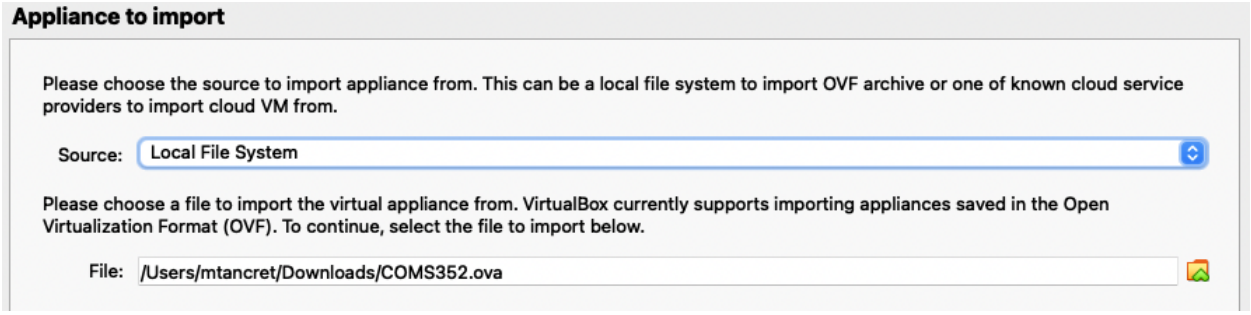
1. Installation

1.1. Installation on a virtual machine

Following this link: https://iastate.app.box.com/s/pytl1mxe9bpemzzp205b9ihjgjyiyk1n, you can find virtual machine images. For Intel based machines download COMS352.ova and for Arm (e.g., Mac M1) based machines download COMS352arm64.utm (download the entire folder).

The .ova image can be run in either VirtualBox or VMWare Workstation Player (these instructions assume you are using VirtualBox). The .utm image can be run in UTM. Install the appropriate virtual machine software for your system now.

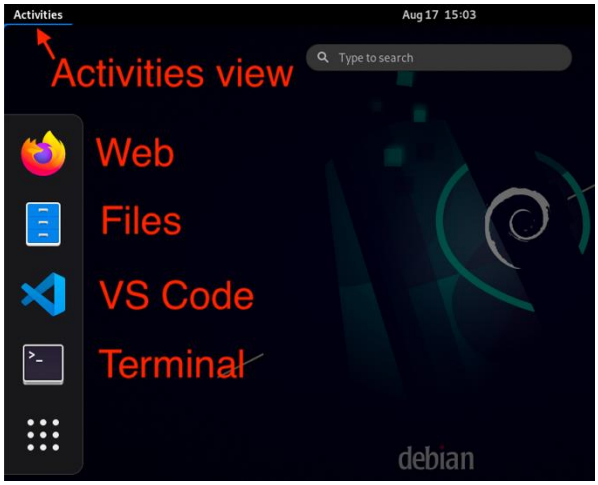
After installing VirtualBox select the menu option File -> Import Appliance to get started.



On the main Manager page select start to begin running the virtual machine.

The password to login the machine is “password”.

Once the desktop has loaded, open applications by selecting the Activities View or use the search bar. The Terminal and VS Code are the main applications you will need to complete this project.



1.2 Installation on pyrite.cs.iastate.edu

In a folder under your home directory, you can clone the git of xv6-riscv and work on it.

To clone:

prompt> git clone https://github.com/mit-pdos/xv6-riscv.git

All the source code will be install in the directory named “xv6-riscv”. In the directory, compile/install the system:

prompt> make

Then, you can use the following command to start the xv6-riscv OS in a virtual machine and enter its shell:

Prompt> make qemu

To quit the virtual machine, press Ctrol+A and then X.

2. The Linux Command Line

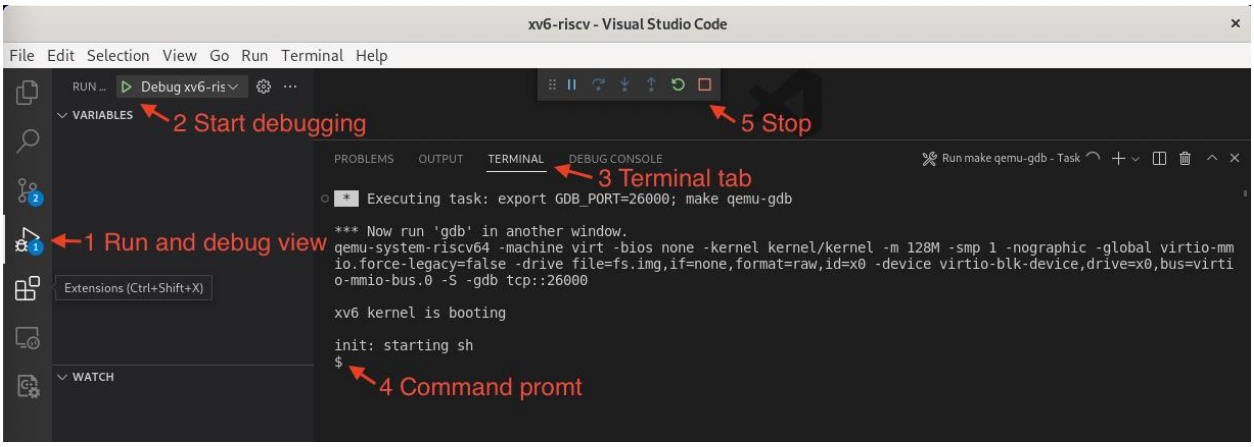
If you are not familiar with Linux command line, here is a nice tutorial:
https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

3. Xv6

3.1. Running Xv6 (from virtual machine)

The xv6 operating system was created at MIT in 2006 to use for teaching Operating System Engineering. It is a modern reimplementation of an early version of Unix, specifically Sixth Edition Unix (or V6) which was originally released in 1975. The reason for using an old version of Unix, is that the code is tiny and simple by modern standards, however it is a real functioning OS that implementations many of the concepts we cover in this class.

To get started, select Activities and open Visual Studio Code. The development environment has already been configured for you. Follow the steps in the image below to start running Xv6.

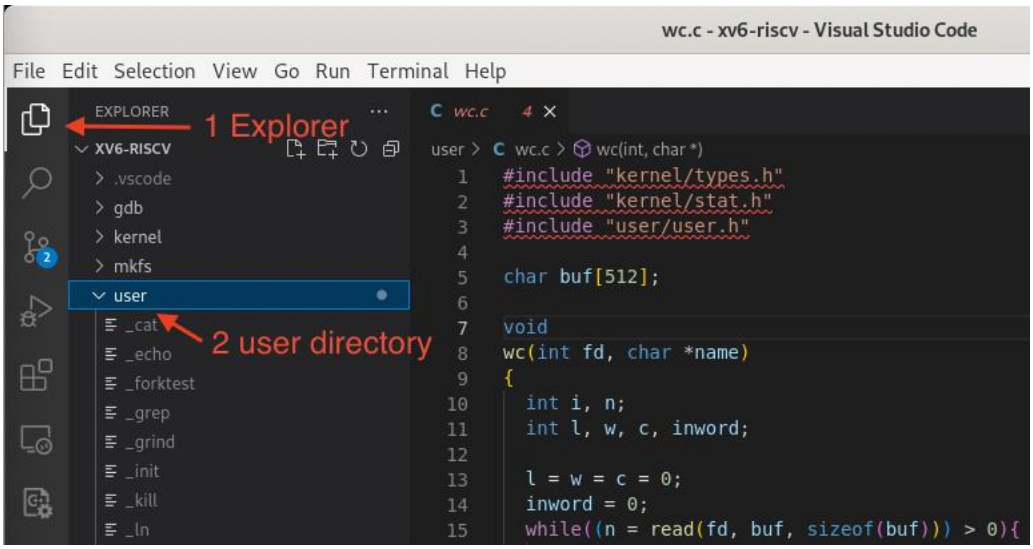


At the command prompt enter a command, for example, `ls`. You should see a list of several files. Note that while Xv6 comes with several Linux-like commands, their features are significantly limited.

3.2. Organization of the Xv6 code

The xv6 source is divided into two folders: `kernel` and `user`. As their names imply kernel is the part of the operating system that executes in kernel mode and user contains several utility programs including a simple shell. In true Unix fashion, commands are not built into the shell, commands are simply stand alone programs. As we dive into operating systems, we will see why this was such an important aspect of the design of Unix.

As an example, we will now explore one of the utility programs, `wc`. You encountered this utility in the command line tutorial. The code is contained in `user/wc.c`.



Warning: you can't include many of the headers commonly found on Linux systems. Only include headers from the xv6 code base. Typically, these 3 are all you need for programs.

```
#include "kernel/types.h"

#include "kernel/stat.h"

#include "user/user.h"
```

Dynamic memory is messy in systems programming (there is no Java Virtual Machine to garbage collect for you), so for simplicity, xv6 almost always uses statically allocated memory. Here we are creating an array to be used as a buffer to hold the incoming stream of text.

```
char buf[512];
```

A single helper function has the responsibility of reading the text from a file descriptor (fd) and printing the counts. We will see that the concept of a file descriptor is an important one, as it allows for a file or the output of another program or many other things to be the source of the text.

```
void

wc(int fd, char *name)

{

 int i, n;

 int l, w, c, inword;
```

```
 l = w = c = 0;
 inword = 0;
```

The read function reads characters from the file descriptor source into the buffer, up to the size of the buffer.

```
 while((n = read(fd, buf, sizeof(buf))) > 0){
  for(i=0; i<n; i++){
   c++;
```

Increment line count if new line character encountered.

```
   if(buf[i] == '\n')
    l++;
```

Reset “inword” if white space character.

```
   if(strchr(" \r\t\n\v", buf[i]))
    inword = 0;
```

Increment word count if new word encountered.

```
   else if(!inword){
    w++;
    inword = 1;
   }
  }
 }
 if(n < 0){
  printf("wc: read error\n");
  exit(1);
 }
```

Output is produced by the printf function. It allows for formatting variables that are put into the output, for example, %d for integer and %s for string.

```
 printf("%d %d %d %s\n", l, w, c, name);
}
```

```
int
main(int argc, char *argv[])
{
  int fd, i;

  if(argc <= 1){
    wc(0, "");
    exit(0);
  }

  for(i = 1; i < argc; i++){
    if((fd = open(argv[i], 0)) < 0){
      printf("wc: cannot open %s\n", argv[i]);
      exit(1);
    }
    wc(fd, argv[i]);
    close(fd);
  }
  exit(0);
}
```