

Project 1A

COM S 352

Fall 2023

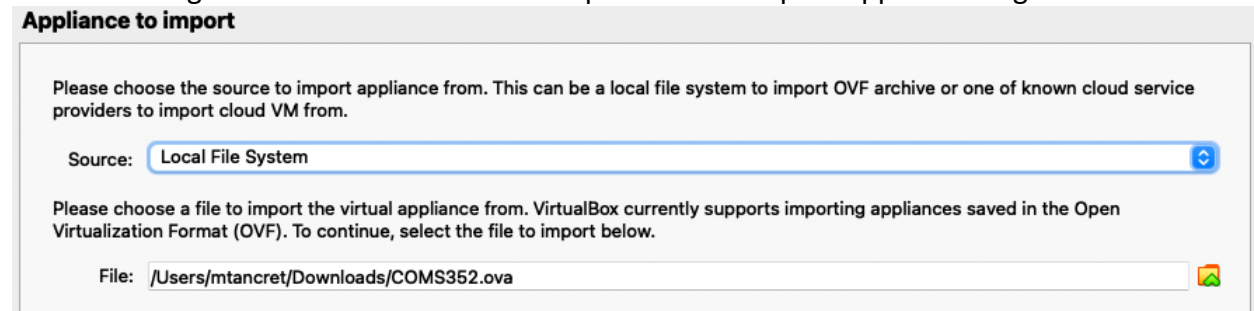
This project is to be completed by yourself. You may ask classmates for help installing software and setting up your development environment. The skills to complete this project are important; you will need them in future projects.

1. Preparation

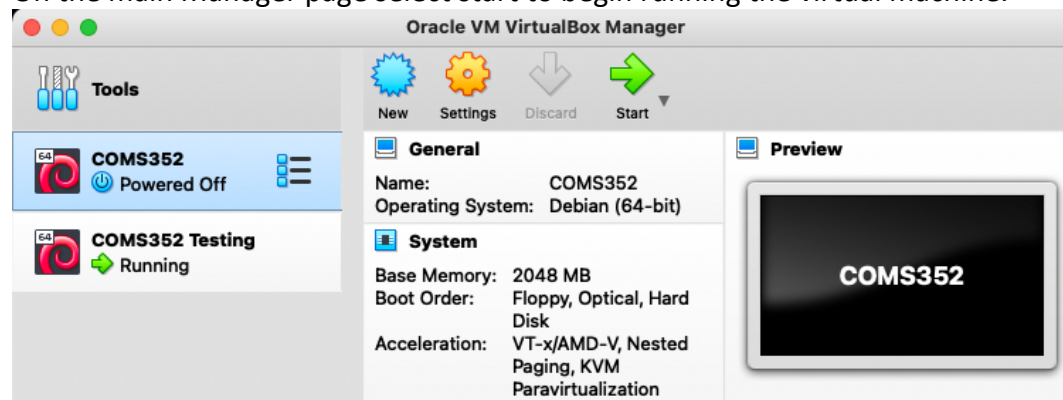
We will be using both the Linux and xv6-riscv operating systems to practice the concepts of this class. You are provided with a virtual machine image linked to on the project page in Canvas. For Intel based machines download COMS352.ova and for Arm (e.g., Mac M1) based machines download COMS352arm64.utm (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.

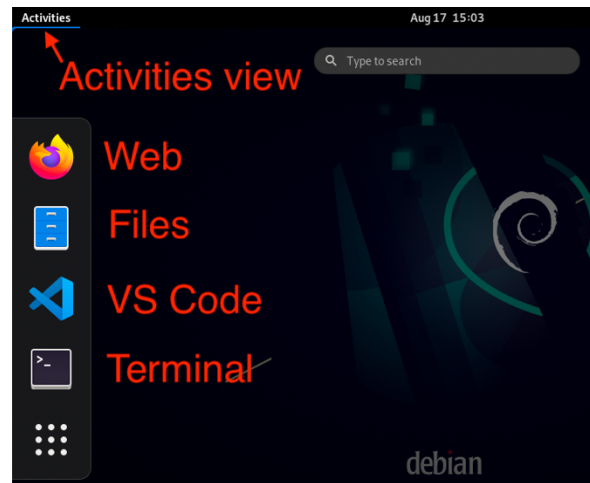
After installing VirtualBox select the menu option File -> Import Appliance to get started.



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



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.



2. The Linux Command Line

The purpose of this section is to become familiar with using the Linux command line. The purpose of this class is to learn how operating system work, however, basic familiarity using the Linux command line will help you work more efficiently on future projects. Complete the Linux command line tutorial: <https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>

To practice what you have learned, complete the following exercise and submit the results.

1. Place the opening paragraph from your favorite book (try <https://www.gutenberg.org/> for access to free books) into a file called `book.txt`.

Hints: You learned in section 4 of the tutorial how to create a new file containing a given text. For copy and paste, click the right mouse button to bring up a menu or use the shortcuts `ctrl+c` for copy and `shift+ctrl+v` for paste.

2. Use command line utilities to create a file called `words.txt` that contains each unique word in `book.txt` on a separate line. The file you create must:
 - a. be sorted alphabetically,
 - b. have all words converted to lower case (so Bob and bob are not counted twice),
 - c. contain no duplicate words,
 - d. contain no forms of punctuation (a word is a string of consecutive letters)

Hint: In addition to the examples you saw in the tutorial, the `tr` command may be helpful, here are some relevant examples of its usage:

From the input file, place each word on its own line.

```
cat input.txt | tr ' ' '\n'
```

Remove all punctuation characters.

```
cat input.txt | tr -d [:punct:]
```

Change all upper case letters to lower case.

```
cat input.txt | tr [:upper:] [:lower:]
```

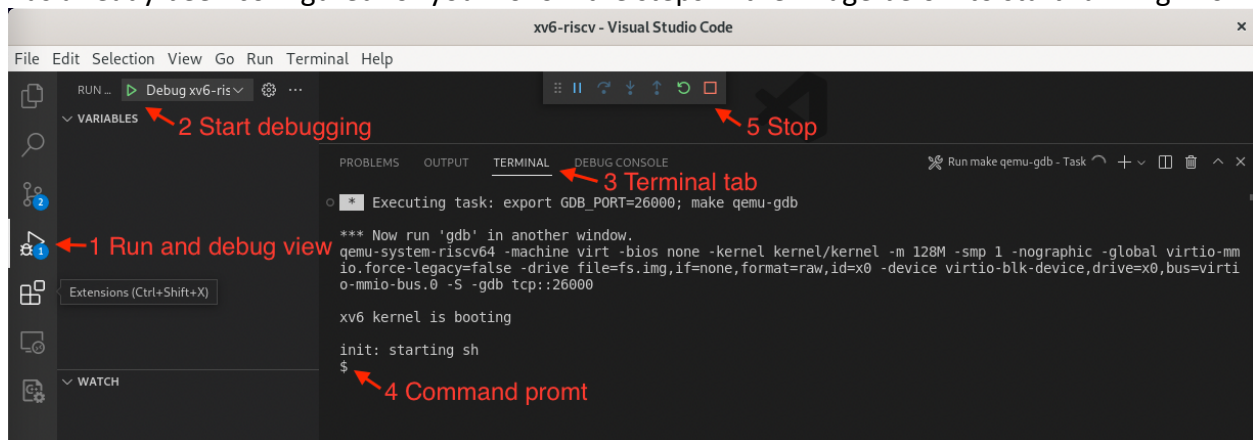
3. To submit your results create a document (Word if fine) that has the original book's text you used, the command(s) you used and the contents of the final output file.

3. Xv6

3.1. Running Xv6

The xv6 operating system was created at MIT in 2006 to use for teaching Operating System Engineering. It is a modern reimplementations 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 implements 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.

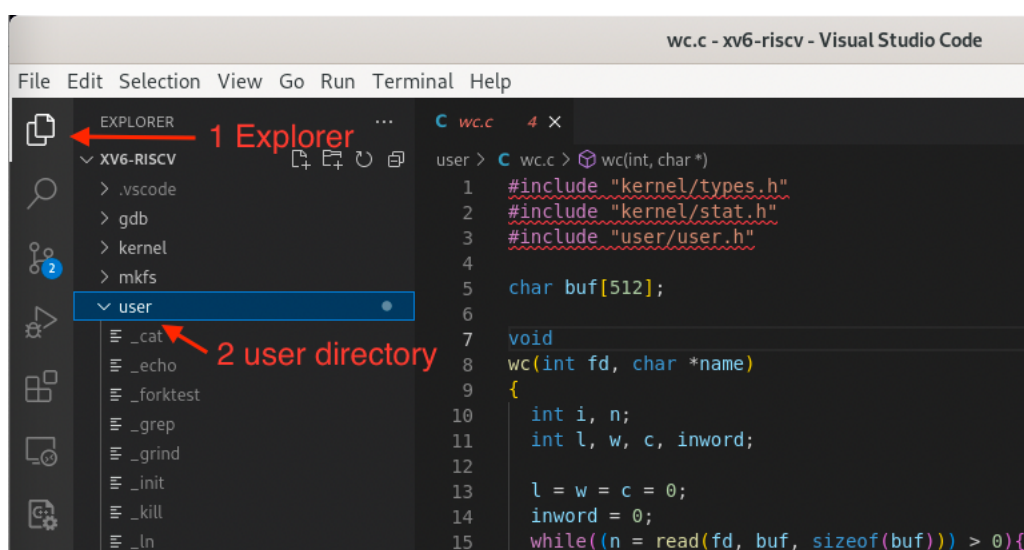


At the command prompt enter a command, for example, `ls`. You should see a list of several files. Note that while Xv6 comes with several Linux-like commands, their features are significantly limited.

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 user 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);
}

```

3.3. Project Task

For this part of the project, modify the `wc` utility to also count the number of vowels (both upper and lower case 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', and 'u', 'U'). The number of vowels must be output as a fourth number, immediately after the number of lines, words, and characters.

Here is the example of using the updated `wc` utility.

```

$ echo "Uniplexed Operating and Computing System" | wc
1 5 43 13

```

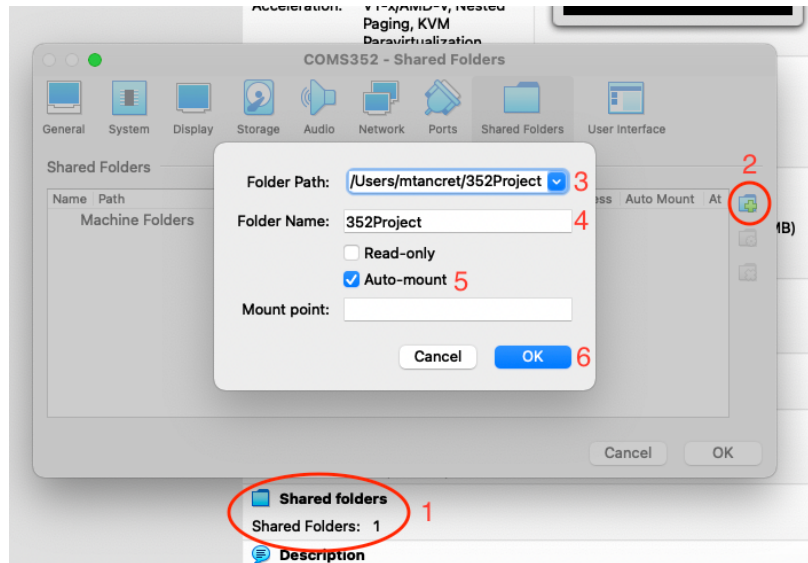
You must document the `wc.c` file, at a minimum add a comment at the top with your **name** and a **description of the changes** you have made.

4. Submission

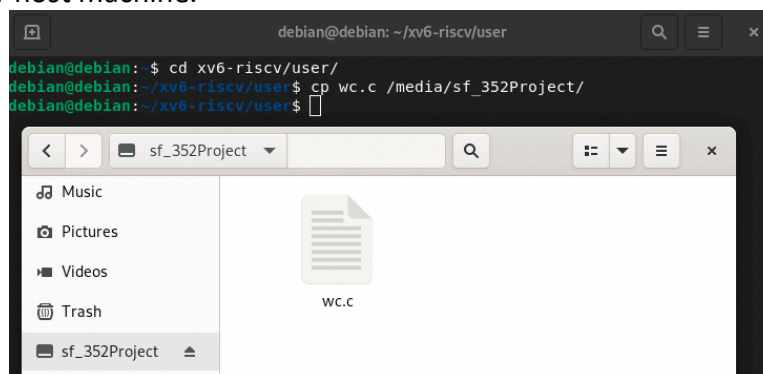
4.1. How to Copy Files out of the Virtual Machine

These instructions are assuming the use of VirtualBox, other virtualization software have a similar way to share files.

The simplest way to move files from the virtual machine to your host machine is by creating a shared folder. Shutdown the virtual machine if it is currently running, and on the main Manager window click on Shared Folders. Navigate to a Folder Path on your where you want to store project files, select auto-mount but keep read-only unselected and mount point empty.



After restarting the virtual machine you will be able to copy files into the directory `/media/sf_folder_name_you_chose`. Those files will be available at the folder path you chose on your host machine.



4.2. What to submit

This project is worth a total of 20 points.

(10 points) As described in Section 2, submit the document describing your results.

(10 points) As describing in Section 3.3, submit the modified code file (along with required comments) `wc.c`. Submit as individual files in Canvas, not in a zip file.