Installing Linux
CMSC 162
Spring 2026

For some of the labs in this course, you will want / need a copy of Linux which you can run on your personal computer or laptop. This is for two reasons:

1. I hope have you use computer graphics in some of the labs, which requires direct access to a computer's drawing hardware. While the systems in the lab are capable of doing this when you log into them in person, these tools don't work (or at least, don't work well) when logged in remotely.

2. While most of our software development will be in the command line using Vim, I am also hoping to introduce you to an "Integrated Development Environment" (IDE). An IDE is a graphical environment for developing software that is often slower to load and use, but which can have many built-in tools that simplify collaboration and software maintenance (debugging).

This handout will walk you through how to:

1. Install Linux to a USB disk or your laptop's hard drive.

Usually this is fairly easy and straightforward. However, some computers need special video drivers or wireless drivers, and this can sometimes cause bizarre issues that take several days to get working. Particularly problematic are systems using Apple's new M1, M2, etc. chips. Apple makes it very difficult (if not impossible) to dual-boot Linux on these systems and students have reported some serious problems getting virtual machines to work in VirtualBox.

2. Use some simple Linux tools (particularly command line tools)

I want you to be comfortable using the environment we use to create software in this course. If you have used a computer by pointing and clicking with the mouse, typing commands into the command line will probably be new to you. It may take you awhile to get the hang of it, but once you do, you will be able to use a computer much faster than ever before.

3. Write, compile, and run software

You will create a simple "hello world" C++ program using the vim text editor and compile it using the gcc compiler.

This program can be reused in your other labs as a starting template so that you don't have to type the same boilerplate code over and over every time you create a new program.

Prerequisites

You will need your computer, a copy of a Linux installation image file (a .iso file), and one of the following:

1. A virtual machine into which you can install Linux (in VirtualBox or VMWare)

2. Enough empty space on your hard drive to create new disk partitions for Linux and set up a "dual-boot" system

3. An empty USB flash drive and a copy of a program like Rufus, "rawrite", or "dd" that can do a "byte-for-byte" copy of the .iso image onto the disk

(Note: You can't simply drag and drop a .iso file onto a flash drive – the .iso file is a byte-for-byte image of a disk filesystem and so it needs to replace the existing filesystem structures, not be a file within them).

Step 2: Obtaining a Linux .iso image file

I recommend Debian Linux, the distribution we use in the Advanced Computing Lab. It is stable, well tested, and easy to use. You can download an .iso file containing an installation image of "Debian Linux" from:

https://www.debian.org/

(then click the "Download" link)

Step 3: Installing Linux

You have three different options for installing Linux on your computer. The easiest is usually to install Linux in a virtual machine running in your existing operating system (Windows, Mac, etc). The virtual machine is usually the simplest option, but it also has some disadvantages:

1. It will run slower than installing to the hard drive

Since the computer is effectively running two complete operating systems, it takes longer (sometimes as much as twice as long) to run programs.

2. It can be much harder to access your files

Your files are inside the "virtual" computer, not on your desktop. There are ways to allow the virtual computer and the host computer to share files, but setting this up is complicated and moving data from one machine to another (for example, if you downloaded code from the

course web site and need to install it on the virtual machine) can be somewhat difficult.

3. The virtual machine takes up a considerable amount of hard drive space. If you don't make the virtual hard disk image large enough, you can run out of space to install programs. There's no easy way to resize it after you've completed the installation. This may not be an issue if you have plenty of free space on your computer and the dual-boot option also has this issue.

4. If the virtual machine image file becomes corrupted, the Linux system may become unbootable and you may lose work.

A second option is to install Linux directly to your hard drive. If you like, you can completely erase your hard drive (deleting all your files) and switch entirely to Linux, but most students prefer to set up a dual-boot system in which Linux and Windows (or other OS) share the hard drive and you can reboot the computer to switch between them.

The dual-boot option will run much faster than the other two options and give you direct access to your files. Once it is set up, it is also usually the safest option. However, if you've never done it before, the installation process can seem complicated – and if you choose the wrong options or something goes wrong, you can make your computer unbootable or overwrite your existing files. I recommend this for advanced users or if you simply can't get a Virtual Machine working.

A third option is to install Linux to a separate external drive (such as a USB flash drive). This is called a "using a live distribution" and it has a lot of drawbacks. **It's very slow**, the external drive is much **more likely to become corrupted** (or lost) than an internal hard drive, and since flash drives tend to be pretty small, you often have **limited space** to save your files. However, the advantage is you can plug the drive into different computers and access Linux from any of them and you can typically use them without touching your hard drive (you MAY still need to alter your boot settings, however).

Option 1: Installing to a virtual machine

If you are running Windows or an Intel-based Macintosh (this is true only of older Apple computers), you can download and install a program from Oracle called "VirtualBox" that will let you simulate a computer system. You can install Linux onto this simulated computer system and run it without making any serious changes to your computer system.

You can obtain VirtualBox here: https://www.virtualbox.org/wiki/Downloads

Under "VirtualBox Platform Packages" select the operating system you typically use. This will download an installer that you can run to set up Virtual Box.

While there IS an option on this page for "Apple Silicon" hosts, students have had mixed results with it. The last time I asked students to install Linux, many of them found that VMWare

worked better. You can download VMWare Player for free (and get a free license for personal use) at https://www.vmware.com/products/workstation-player.html

Option 2: Installing to your hard drive

If you install Linux to your hard drive, the install program will insert some code called a “boot loader” into the boot sequence of your computer. Every time you reboot the computer, you will see a menu that lets you pick which operating system to boot. In order to switch operating systems, you will need to shutdown or reboot your computer each time.

If you wish to use this option, the first step is to create a live disk (as below) and then configure your system to boot from it. An installer will run that will guide you through the process. You will need to be very careful in the hard drive partitioning step to make sure you don’t accidentally overwrite your files.

If you run Windows 10 or Windows 11, I strongly recommend partitioning your drive using Windows before installing Linux. That way, Windows will be aware of the change and won’t “freak out” the first time you boot it after the Linux installation.

Note: If you decide to install Linux to your hard drive, you should back up all your files first.

Option 3: Creating a live disk

One of the neat things about Linux is that you can try it out without installing it on your computer using something called a "Live disk". To use the Live disk, you just put it into your computer and reboot. This works on most Windows computers, but doesn’t work so well on a Macintosh.

In some cases, you have to first configure your computer to boot from an external drive. You can do this in the BIOS or UEFI settings. On some Windows computers, you have to do this from the “Advanced Startup” settings. However, some computers also allow you to enter the BIOS settings when the computer boots by holding down a particular key or combination of keys.

On a Dell, you usually get to these settings by holding down the F2 key while you turn on your computer. On Levono laptops, you may need to press Enter and then F1. On other systems, there may be a different set of keys (such as DELETE). Pay attention to the screens that come up when you first boot your computer – they usually tell you the proper key combination.

Step 4: Booting Linux

In a virtual machine

If you have installed Linux to a virtual machine, you should be able to pull up Virtual Box and double click on the virtual machine to start it.

From the hard drive

If you have installed Linux to your hard drive in a dual-boot configuration, you will need to reboot the computer to start it. When the boot menu comes up, use the cursor keys to select the option for Linux and press enter.

From a live disk

If you are using the Live DVD or Live USB method, insert the DVD or USB thumb drive into your computer and reboot. When the boot menu comes up, use the cursor keys to select "Start Kubuntu" and press enter. Then wait. It can take anywhere from one to five minutes for Linux to boot off the live disk, so be patient.

If the boot menu does not appear, you probably need to adjust your UEFI / BIOS settings.

Linux is usually much faster than Windows, but most USB drives are very slow (and even if the drive is fast, the USB port it connects to may not support high speed access). Many USB flash drives transfer data at about 25MB/s, much slower than a hard drive or SSD. So – be patient. Fortunately, once the boot disk has loaded most of Linux into the computer's memory, it will stay there. That means that after a slow initial boot, the system should run acceptably fast (but probably still slower than you're used to).

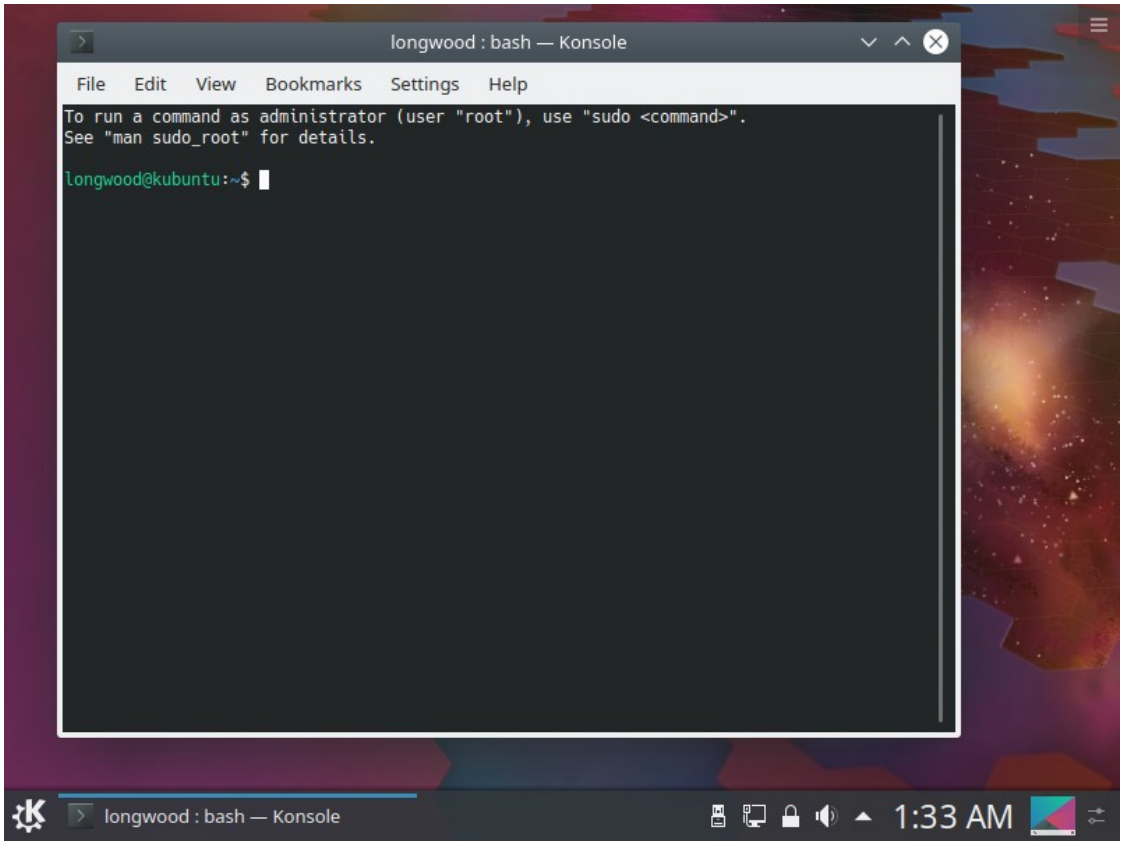
Accessing a terminal

Once the system boots, it will probably prompt you for a username and password. Type in the values you set during the software installation process. If you are booting from a live disk, the system may be set up to launch without a password. For security, it’s a good idea to set one – feel free to ask me for help if you need to do this.

After you log in, you should see a Linux desktop environment. A desktop environment provides menus, windows, buttons, icons, sliders, and all the other graphical controls that let you interact with software. Unlike Windows or Mac OS X, Linux has many different desktop environments that you can choose from. They lay out the desktop in different ways. Some of them may have a system tray at the bottom of the screen, while others may use an autohiding dock that appears and disappears when you move the mouse to one side of the screen. There are many other differences between the options.

The most popular desktop environment choices are Gnome, KDE Plasma, and XFCE. Which one of these is the default depends on which flavor of Linux you installed. In most systems, the login screen has a drop down menu that lets you switch between desktop environments. You may want to play around to find which one you are most comfortable using.

Here is what the KDE Plasma environment looks like with a “terminal” program running in the foreground and several background processes running in the system tray.



(Note: Any weird dots are an artifact of how I created this screenshot – please ignore them)

Step 5: Opening a terminal

Depending on your desktop environment, there may be many different ways to launch a program. For example, in KDE Plasma and XFCE, you can usually pull up an Applications menu by clicking on the icon in the bottom-left corner of the screen. In Gnome, you can hover over the applications bar and click on the icon that looks like a grid of small squares.

Most desktops will also allow you to type ALT-F2 or ALT-SPACE to pull up a launcher that lets you type in the name of a program. Try to figure out how to launch the program named “konsole” (or possibly “Konsole Terminal”).

In KDE Plasma, if you find the terminal program in the menu, you can right click on the icon for it and select "Add to favorites" or "Add to panel." This will make it a lot easier to pull up the terminal in the rest of this lab, which I highly recommend.

The terminal gives you a **command-line interface** to Linux. It will display a prompt that looks something like:

```
longwood@localhost:~$
```

This prompt is the computer's way of telling you that it is waiting for you to type a command. In this example, it tells you that you are logged on as user "longwood" on the computer named "localhost" and that the folder you are currently in is the "~" folder (the ~ symbol is a shorthand for your home directory, which we'll talk about in a moment). You probably set a different username when you installed Linux, so instead of "longwood" you likely have something different. If you gave your computer a name during the install process, you may see that name instead of "localhost".

If you click on the terminal window with your mouse, the operating system will give that window the "focus". Any keystrokes you type will be sent to the terminal's shell program as a command.

We can now enter some simple Linux commands and see what they do. Some of the most important commands are:

`ls` – List the files in the current folder

`more` *`filename`* – Display the contents of a file one page at a time

`vim` *`filename`* – Edit the file named *filename*

Another very important commands is the command to install a new piece of software.

Unfortunately, the install command is different on different Linux distributions. For example, in **Debian**, Ubuntu, and Mint, the command to install g++ looks like this:

```
sudo apt install g++
```

However, on OpenSuSE, it looks like this:

```
sudo zypper install gcc-c++
```

And on Archlinux:

```
sudo pacman -S base-devel
```

On Fedora, Redhat, and CentOS, it looks like:

```
sudo yum install gcc-c++
```

Notice that the name of the package to install is sometimes also different in one Linux distribution than in another. (Usually it's either the same or very similar, though)

In this class, two programs you will need are "g++" (the C++ compiler) and "vim" (an editor).

Install g++ and vim now

Step 7. Installing SDL

Many of the examples in the textbook rely on the SSDL library written by the author. Let's download a copy of that library and install it. The easiest way to do that is to type:

```
cd ~/
wget http://marmorstein.org/~robert/Spring2022/cs160/SSDL.tar.gz
```

When the download is complete, open up a terminal window and type:

```
tar xvf SSDL.tar.gz
```

When this command has finished, you should have a copy of the SSDL library in your home directory.

If you type "ls", you should now see a folder named "external" in your home directory.

Earlier in this lab, you learned how to install packages in Linux using the apt command. Let's practice that now. Install these packages (note libsdl2 only has one digit in it (the 2). The lowercase L right before the 2 looks like a number 1, but it's not):

```
libsdl2-dev
libsdl2-gfx-dev
libsdl2-image-dev
libsdl2-mixer-dev
libsdl2-net-dev
libsdl2-ttf-dev
```

SSDL also needs us to download some Microsoft fonts. Unfortunately, this requires some extra work. First, we need to enable the correct repositories in Debian to access these files.

To do that, run these commands:

```
sudo sed -i 's/main/main contrib non-free/g' /etc/apt/sources.list
sudo apt update
```

Each time you type a "sudo" command, you may be prompted for your password. That's normal.

Second, the way these are licensed requires you to agree to a license file before they can be installed. You may or may not get a screen asking if you agree with license terms. If this shows up and you do it incorrectly, it can be very difficult to fix, so please read these instructions carefully: when the screen appears, press the "Tab" key (to move the cursor to the "accept these terms" button) and THEN hit "Enter".

The font packages are named "ttf-mscorefonts-installer" and "fonts-liberation". The Liberation fonts are free fonts, not Microsoft fonts, but we need both. Install them now with apt.

Now that we have all the libraries installed, we can compile SSDL itself. Type (press enter after each command):

```
cd external/SSDL/unix
make
```

This should run some compile commands. When it finishes, SSDL should be built. If you type "ls" and press enter, you should see several files listed and one of them should be "libssdl.so". If not, something is wrong (and you should probably ask for some help with this step).

Now type:

```
vim ~/.bashrc
```

Scroll to the bottom of this file. Type "o" to get into insert mode, then type:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/external/SSDL/unix
```

This changes a setting, but will only take effect when you log in to Linux. You could log out and back in – or you can use the following command to run the script manually (be sure to quit vim first!):

```
source ~/.bashrc
```

SSDL is now installed and configured!

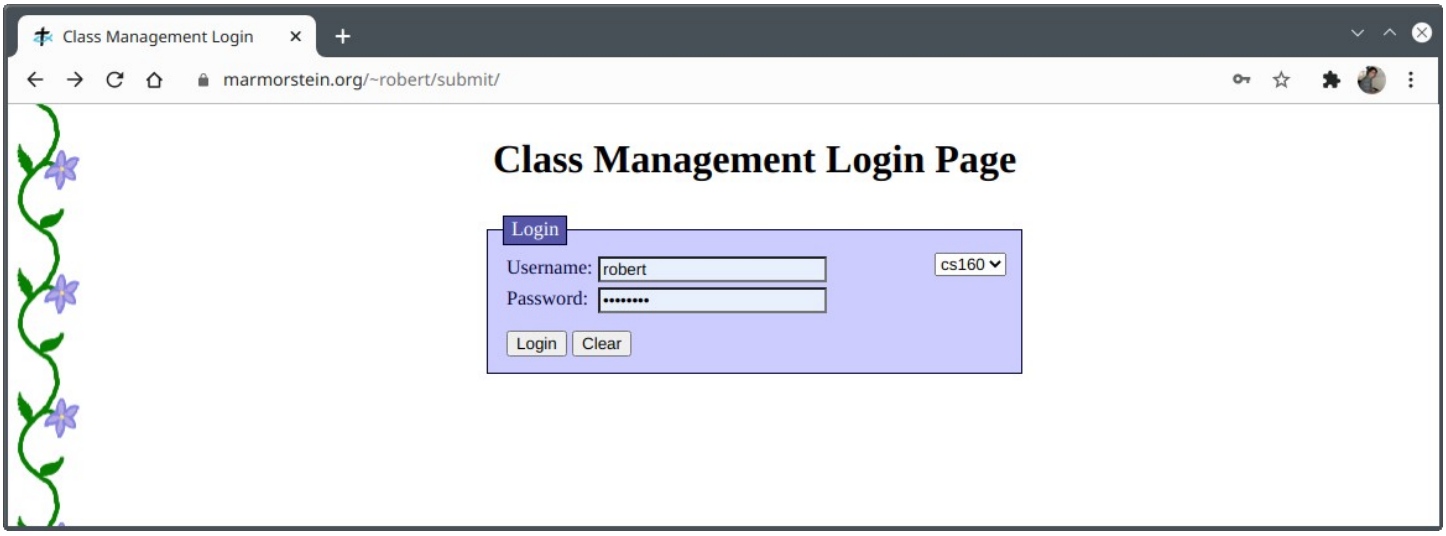
You can download a Makefile from SSDL project from the course web site. You can copy this file into your project folder and then type "make" to compile and run SSDL projects.

```
wget https://marmorstein.org/~robert/Spring2022/cs160/Makefile
```

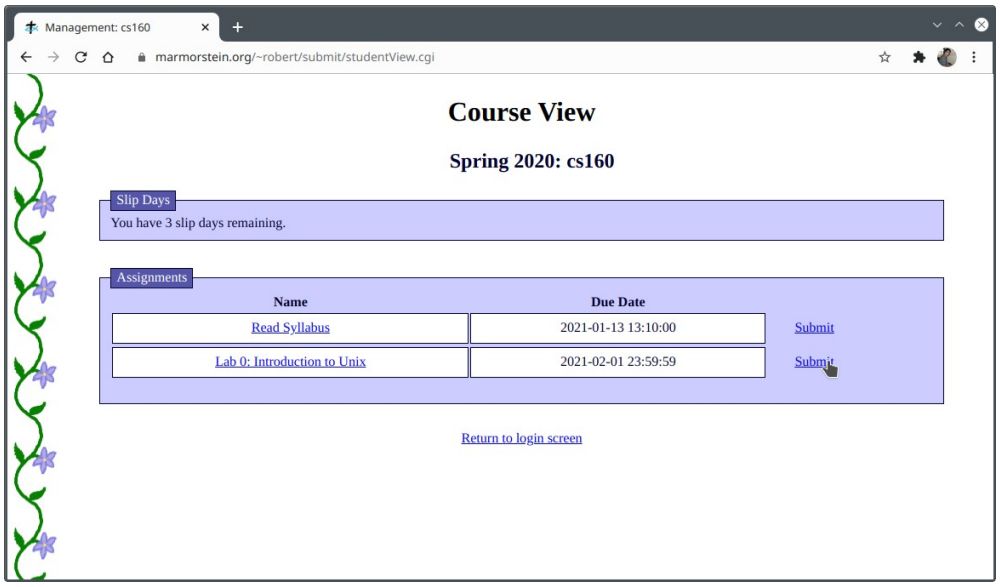
Note: Submitting Projects

To submit a project for this course, open up a web browser and browse to http://marmorstein.org/~robert/submit/

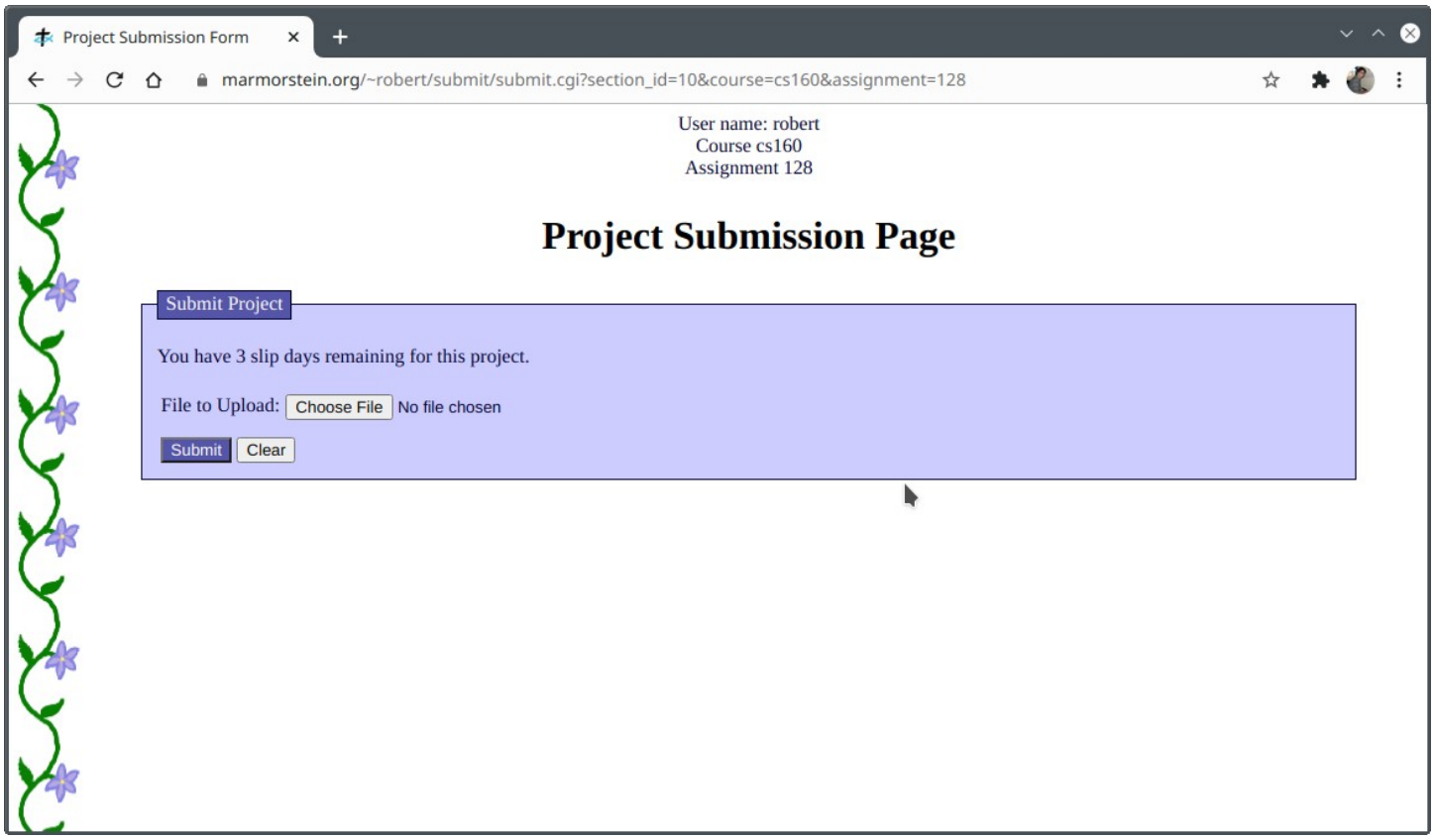
Use the credentials you received in class to log in to the submit page for the cs162 course:



Then find the correct assignment in the list and click the "Submit" link to the right of it:



Click "Choose File" and browse to your **LabX.tar.gz** file. Open it and then click "Submit" to upload your project.



Wait until you see a page confirming that you have successfully submitted and then close the browser window (if you close the window before the upload has finished, I may get only a partial submission).

Congratulations! You have submitted your first project for this class.