

## Lab 1

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, you can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstance, contact your lab TAs and Jennifer Parham-Mocello.

**Pair Programming:** In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: [Pair Programming Student Handout](#)

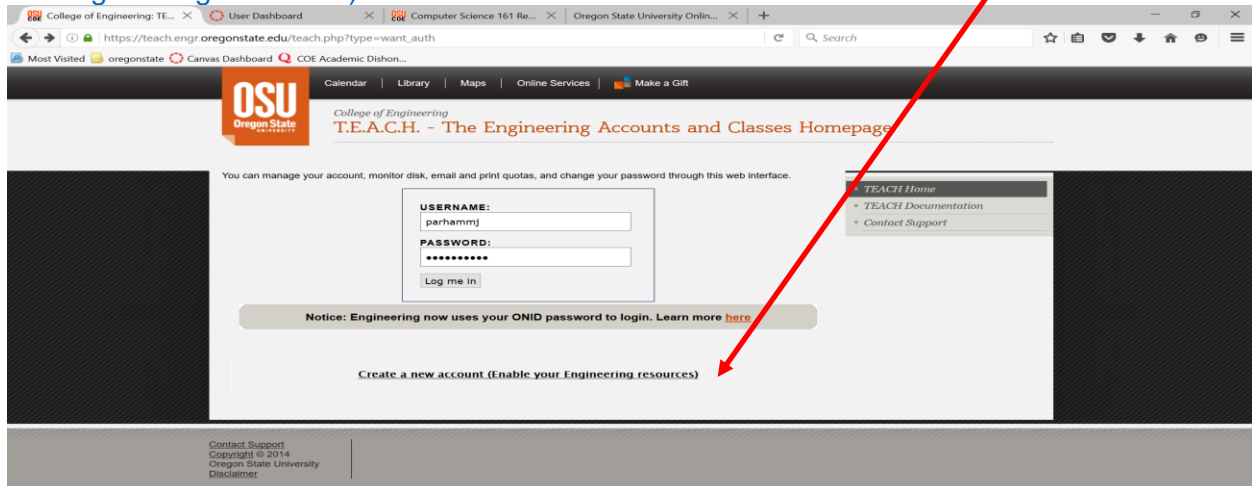
**(2 pts) Icebreaker:** Believe it or not, we do have to communicate with others being a computer scientist and electrical engineer, and others have to communicate with us.☺ Let's get to know our peers. Get into small groups of 4-5, and answer the following about everyone in the group:

- Do you have any experience programming in C/C++? If so, how long?
- What did you like (or dislike) the most about your Summer 2017?
- “BE PROACTIVE” is one of our class mottos this quarter. Being proactive, establish one goal for yourself this Fall quarter.

### (2 pts) Getting Set Up

These **exercises need to be completed individually by each student.** This is to ensure that everyone gets setup properly! You can certainly ask your neighbor or friend questions! ☺

1. **If you do not have an ENGR account**, then you will need to create one by clicking on the link at the bottom of the login page that says: [Create a new account \(Enable your Engineering resources\)](#)



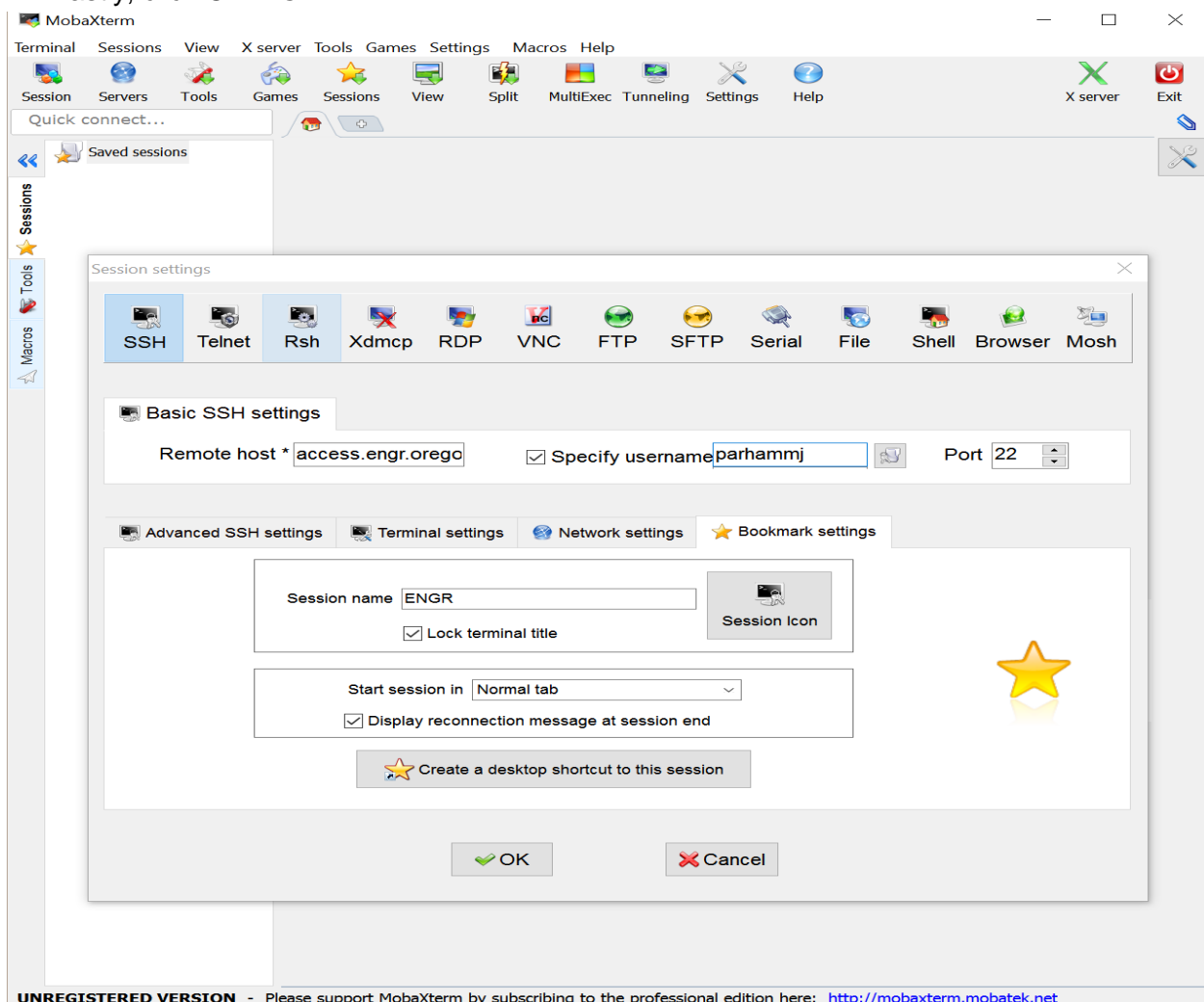
2. **Windows Users:** Download a free Secure Shell (ssh). We will use the MobaXterm this quarter. If you already have Putty installed, this is fine too!!!  
MobaXterm: <http://mobaxterm.mobatek.net/download-home-edition.html> (Use Installer Edition, unless you are not using your own computer, and extract files in zip to not get a bunch of messages about accessing files.)

**Mac or Linux Users:** You have a terminal w/ ssh built into the OS. Open the terminal, and at the prompt, type  
**ssh username@access.engr.oregonstate.edu**

3. **Windows Users:** Once you have an engineering account and installed MobaXterm, open MobaXterm. Go to **Sessions -> New Session**, and click on the **SSH icon**.

Enter **access.engr.oregonstate.edu** as the remote host, and click on the **Specify Username** checkbox to enter your username in the appropriate field.

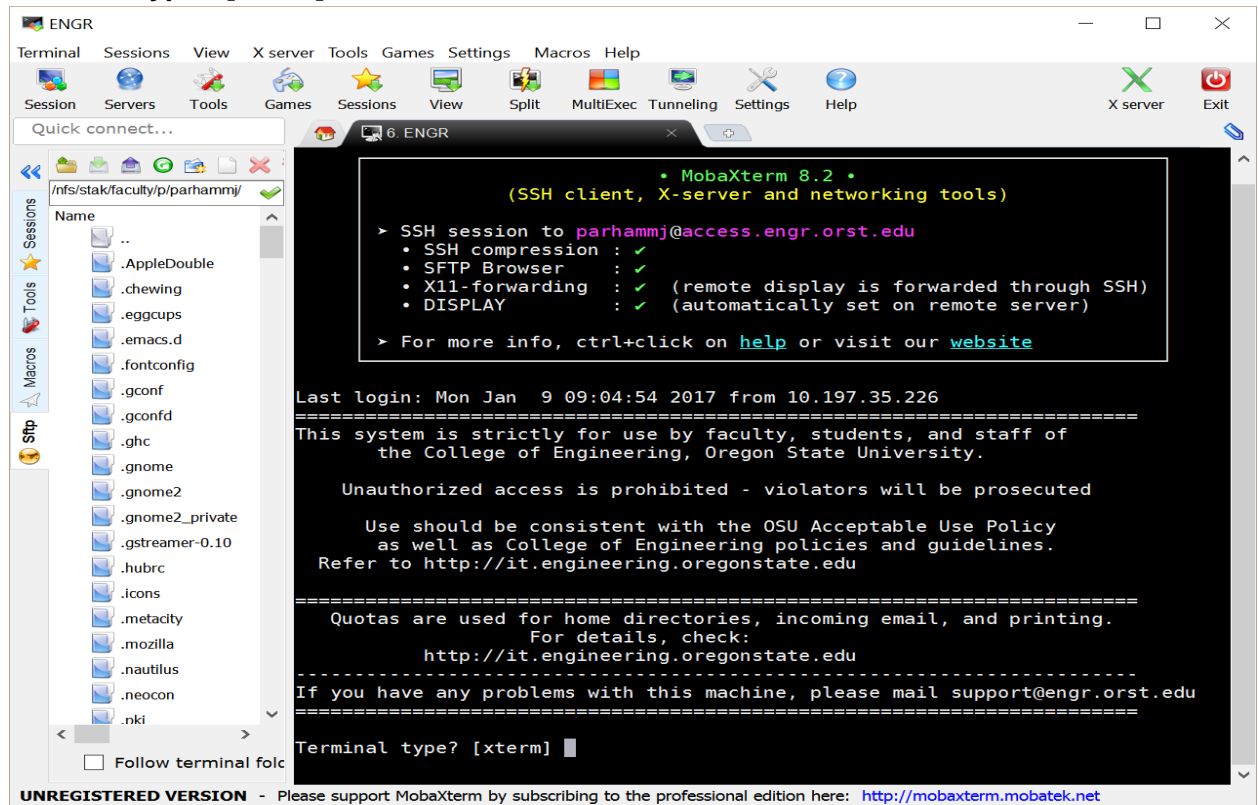
Click on the **Bookmark settings** tab to save your session with a specific name, e.g. ENGR. Lastly, click OK!!! 😊



4. The ENGR session you just created will connect, and you will be asked to enter your password at the prompt. **\*\*Note:** You will not see anything as you type your password. This is a security feature in Linux!!!

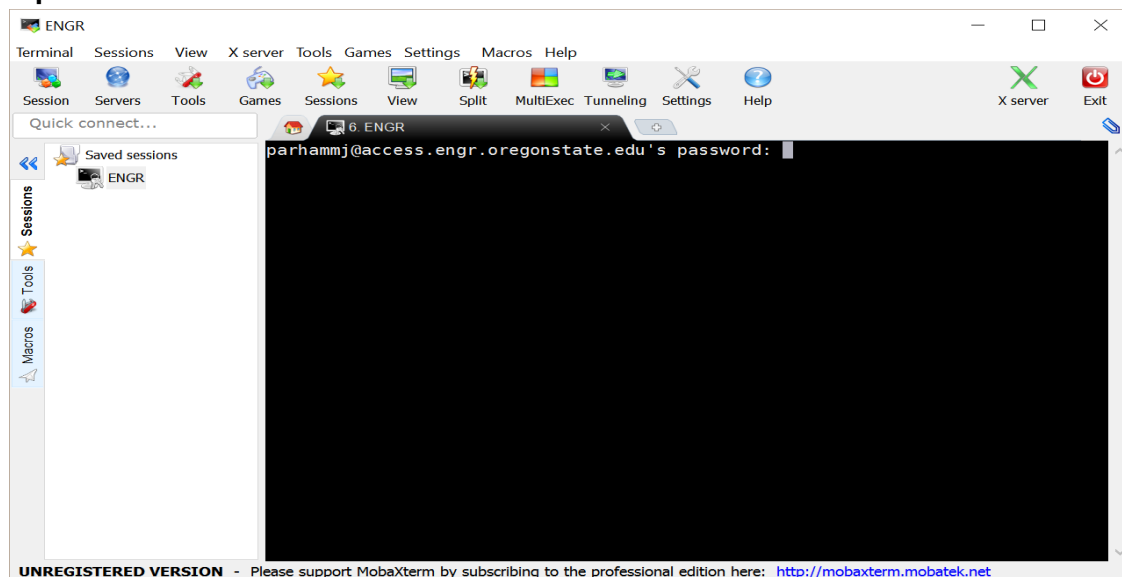
After a successful logon, press enter at the prompt below:

**Terminal type? [xterm]**



Then you get a prompt that looks something like this:

**flip1 ~ 1%**



5. At the prompt, type the following commands to look at your files and directories in your home directory. Note the differences to your lab instructor.

**ls**

**ls -a**

**ls -l**

**ls -al**

**\*\*Note:** You should notice a `.` and `..` directory listed. The `.` directory refers to your current directory, and the `..` directory refers to one directory above your current directory. The `~` refers to your home directory.

6. UNIX/Linux provides you manual pages for the commands. You are encouraged to read these manual pages when you have questions about a specific command or want more details about the options to use with a command. Use the **space bar** to scroll forward through the manual pages (one page at a time), **press b** to scroll backwards (one page at a time), and **press q** to quit the manual page. You can also use the up and down arrow keys to scroll forward and backward one line at a time, but who wants to do this 😊

**man ls**

7. In addition, if you are not sure what a command is in UNIX/Linux, then you can find the appropriate command using `apropos` and a keyword. For example, what are the UNIX/Linux commands for editing a file, working with a directory, etc.

**apropos editor**

**apropos directory**

8. You may have noticed that you get more text than what can fit in your terminal window. To view data a page at a time in your terminal, you can pipe the command contents through another command called `less`. This will allow you to scroll through the pages using **space bar, b, and q** just as you did with the manual pages.

**apropos directory | less**

## **(2 pts) Create Directories, Write C++ Program, & Answer Metacognitive Questions**

9. Make a directory in your home directory named `labs`, and change into the `labs` directory.

**mkdir labs**

**cd labs**

10. Create a directory in your `labs` directory named `lab1`, and change into the `lab1` directory.

**mkdir lab1**

**cd lab1**

11. Make a note of your present working directory.

**pwd**

12. You can go back/up a directory by using two periods/dots together, and you can go back to your home directory by using the tilde, `~`. Use the `pwd` to confirm you are back in your home directory.

**cd ..**

```
cd ~  
pwd
```

13. Now, change into the labs/lab1 directory by using your **up arrow key** to take you through the history of commands you've used in the past. You should see the **cd labs** and **cd lab1** command you typed earlier. You can also change directly into the lab1 directory by using **cd labs/lab1**.
14. A good rule of thumb is to use **pwd** at any time to determine where you are, in case you forget☺ Also, don't be scared to use **ls** as often as you need to see a listing of your current directory!
15. Use the vim editor to create a C++ file containing your first C++ program.  
**vim hello.cpp**

Below is an overview of vi/vim and a set of useful commands. \*\*Remember you can **man vim** to see the manual pages for vi/vim. In addition, I have provided a useful vim tutorial under **Useful Links on our class website**. One of the default editors that come with the UNIX operating system is called vi (**v**isual editor) or vim (**v**isual editor **i**mproved). [Alternate editors for UNIX include emacs and pico.]

The UNIX vi editor is a full screen editor and has two modes of operation:

- *Command mode* commands which cause action to be taken on the file, and
- *Insert mode* in which entered text is inserted into the file.

In the insert mode, every character typed is added to the text in the file; pressing the <Esc> (*Escape*) key turns off the Insert mode.

#### **Basic Commands (in command mode):**

**:w**<Return> - write out modified file to file named in original invocation  
**:wq**<Return> - quit vi, writing out modified file to file named in original invocation  
**:q**<Return> - quit (or exit) vi  
**:q!**<Return> - quit vi without saving the latest changes  
**:0**<Return> - move cursor to first line in file  
**:n**<Return> - move cursor to line n  
**:\$**<Return> - move cursor to last line in file  
**:set number**<Return> - insert line numbers into vi file

**/search**<Return> - find first occurrence of search from the location of cursor in the downward direction

**?search**<Return> - find first occurrence of search from the location of cursor in the upward direction

**n** - move cursor to next occurrence of last search (**in direction of search**)

**j** [or down-arrow] - move cursor down one line

**k** [or up-arrow] move cursor up one line

**h** [or left-arrow] move cursor left one character  
**l** [or right-arrow] move cursor right one character  
**0** (zero) - move cursor to start of current line (the one with the cursor)  
**\$** - move cursor to end of current line  
**^** - move cursor to the first non-whitespace character in a line  
**w** - move cursor to beginning of next word  
**b** - move cursor back to beginning of preceding word  
**u** – undo whatever you last did  
**x** – delete current character  
**dd** – delete current line  
**yy** – yank line and put into buffer for pasting  
**p** – paste text in buffer to line below cursor  
**i** – enter insert mode and enter text before the cursor  
**a** - enter insert mode and append text after cursor  
**o** - enter insert mode and enter text on line below cursor  
<Esc> - get out of insert mode and enter command mode

16. Write the infamous “hello world” program as your first piece of C++ code. Use the style guideline on the class website for suggestions on how to format your code:

[http://classes.engr.oregonstate.edu/eecs/fall2017/cs161-001/161\\_style\\_guideline.pdf](http://classes.engr.oregonstate.edu/eecs/fall2017/cs161-001/161_style_guideline.pdf)

```
#include <iostream>

int main() {

    std::cout << "Hello World!" << std::endl;

    return 0;
}
```

17. Compile and execute your C++ “hello world” program.

**g++ hello.cpp -o hello**  
**./hello**

For each question, make a change to the hello.cpp file, and recompile (**g++ hello.cpp -o hello**) and possibly execute the program (**./hello**) if there are no errors to answer the question

**Question 1: What happens when you forget the semicolon at the end of the cout statement? Be specific.**

```
std::cout << "Hello World!" << std::endl
```

**Question 2: What happens when you forget to include iostream, i.e. remove `#include <iostream>` from the program? Be specific.**

**Question 3: What happens when you remove the `<< std::endl`? What is the difference in having it and not having it?**

```
std::cout << "Hello World!";
```

**(2 pts) Add a variable and reading input to the program:**

18. Now, let's add a variable (or a place to store information) in our program. We will create a place to store an integer (or whole number). Then, we'll ask/prompt the user to enter a number, read the input from the user, and display it back to the screen.

```
#include <iostream>

int main() {
    int number;

    std::cout << "Hello World!" << std::endl;

    std::cout << "Enter an integer whole number: ";
    std::cin >> number;
    std::cout << number << std::endl;

    return 0;
}
```

19. Compile and execute your C++ "hello world" program again with the new changes.  
**g++ hello.cpp -o hello**  
**./hello**

20. Logout of the remote machine by typing  
**exit (or logout)**

For each question, make a change to the hello.cpp file, and recompile (**g++ hello.cpp -o hello**) and possibly execute the program (**./hello**) if there are no errors to answer the question

**Question 4: What is the difference between the first and second program you wrote?**

**Question 5: What happens if you mix the direction of `>>` with the `cin`? Be specific.**

```
std::cin << number;
```

**Question 6: Why do you need to create a variable to display a number entered by the user?**

## (2 pts) Upload Written Document and Program (.cpp) to [TEACH](#)

Since you have to upload your assignments from your local computer or the engr server to the [TEACH](#) website, then we should practice now ☺ It is easy to upload files from your local computer to TEACH, as long as you remember where you save it!!! ☺ However, it is not as easy for you to upload files from the ENGR server to TEACH.

### Transfer Files

You can transfer the file to your own computer, then upload it on TEACH:

**Windows Users:** MobaXterm already has a sftp client installed. Notice the window pane on the left with folders and files. You can drag and drop a file to your local desktop to select on TEACH.

**Mac OS Users:** Download Cyberduck from <http://oregonstate.edu/helpdocs/osu-applications/offered-apps/ftp>

**Upload your 6 answers to the questions and hello.cpp file (not the executable) to TEACH!!!!**

The screenshot shows the MobaXterm application window. The left sidebar contains a file explorer for an SFTP session to `/nfs/stak/faculty/p/parhammj/161/`. The main terminal window displays a login banner for the ENGR system, including a warning about unauthorized access and a list of users.

**Terminal Output:**

```
Last login: Mon Jan 9 09:04:54 2017 from 10.197.35.226

=====
This system is strictly for use by faculty, students, and staff of
the College of Engineering, Oregon State University.

Unauthorized access is prohibited - violators will be prosecuted

Use should be consistent with the OSU Acceptable Use Policy
as well as College of Engineering policies and guidelines.
Refer to http://it.engineering.oregonstate.edu

=====
Quotas are used for home directories, incoming email, and printing.
For details, check:
http://it.engineering.oregonstate.edu

=====
If you have any problems with this machine, please mail support@engr.orst.edu
=====

Terminal type? [xterm]
09:07:27 up 10 days, 11:45, 10 users, load average: 1.20, 1.26, 1.35
USER  TTY  FROM          LOGIN@  IDLE   JCPU   PCPU   WHAT
olivasd pts/1  10.197.36.130 08:58   3:35   0.10s  0.10s -tcsh
hillyers pts/3  76.105.238.185 Fri22   2days 0.09s  0.09s -tcsh
satherde pts/4  71.80.154.81  06:02   1:19m  4.11s  3.96s vim
yapdo pts/5  :1           Wed22   4days 0.00s  1:01  kdeinit4: kded
liale pts/6  10.197.36.16  05:03   4:02m  0.15s  0.15s -bin/tcsh
yapdo pts/7  :1           Thu21   3days 0.07s  0.07s -bin/tcsh
parhammj pts/8  10.197.35.226 09:06   7.00s  0.22s  0.04s w
zhaoshi pts/9  73.240.248.56 Sat21   35:09m 0.28s  0.09s ssh zhaoshi@ri
rugliokm pts/11 10.197.35.179 08:39   19:59  0.90s  0.16s sshd: rugliokm
setok pts/30 210.147.192.206 22:04   7:50m  0.46s  0.46s -tcsh
flip3 ~ 151%
```

**File Explorer (Left Sidebar):**

- 2-d.cpp
- 2-d\_structs.cpp
- a.out
- address\_arith.cpp
- addresses.cpp
- args.c
- args.cpp
- args2.cpp
- arrays.cpp
- dasses.cpp
- dasses\_in\_classes.cpp
- compilers.c
- continue\_test.cpp
- ctest.cpp
- exam.cpp
- functions.cpp
- more\_arrays.cpp
- new.cpp
- nass arravs.cnn

**Footer:** UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>



**AND**

### **Map Network Drive**

2). You can map a network drive, and choose the file from the mapped network drive. Here is some software to help you with this.

Lastly, you want to map a network drive to the ENGR server. This allows you to directly work off the server as if it were a disk drive on your computer. You can follow these instructions to map a network drive for Windows or MacOS.

#### **Windows:**

<http://it.engineering.oregonstate.edu/accessing-engineering-file-space-using-windows-file-sharing>

#### **MacOS:**

<http://it.engineering.oregonstate.edu/accessing-engineering-file-space-using-os-x-smb-command>

**If you want to use the drive off campus**, then you must download the **Cisco VPN Client** from OSU: <http://oregonstate.edu/helpdocs/osu-applications/offered-apps/virtual-private-network-vpn>

**\*\*NOTE:** You may want to put these programs on a flash drive to carry with you in your backpack. This will help you get around from any computer without needing your laptop all the time.