

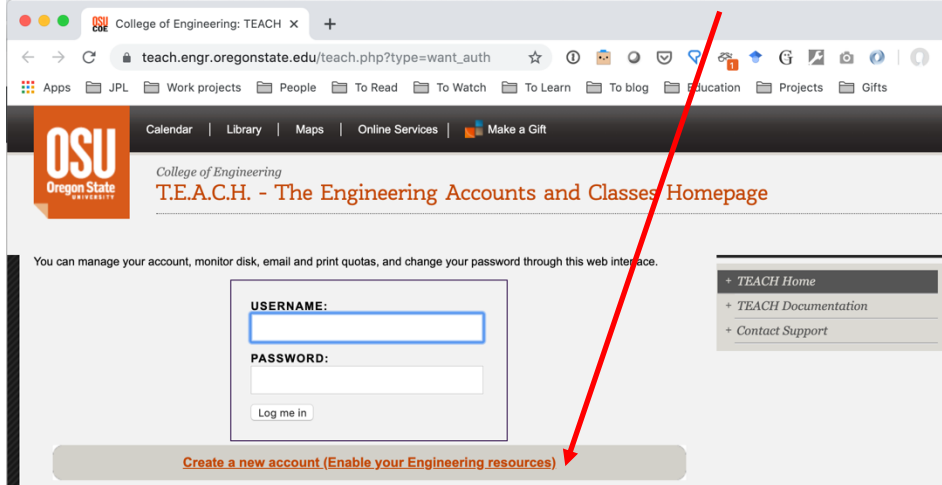
CS 161 Lab #1 – Off to a Good Start

To get credit for this lab, you must be checked off by a TA by the end of lab. For non-zero labs, you can earn a maximum of 3 points (of 10) for lab work completed outside of lab time, but you must finish this lab before the next lab begins. For any exceptions, contact your lab TAs and Dr. Wagstaff in advance.

(2 pts) A. Get 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, friend, or TA questions! ☺

1. Log into TEACH to see if you have an ENGR account. **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. **Log on to the ENGR server through a secure shell.** Depending on your operating system, jump to the instructions for [Windows](#) or [MacOS/Linux](#).

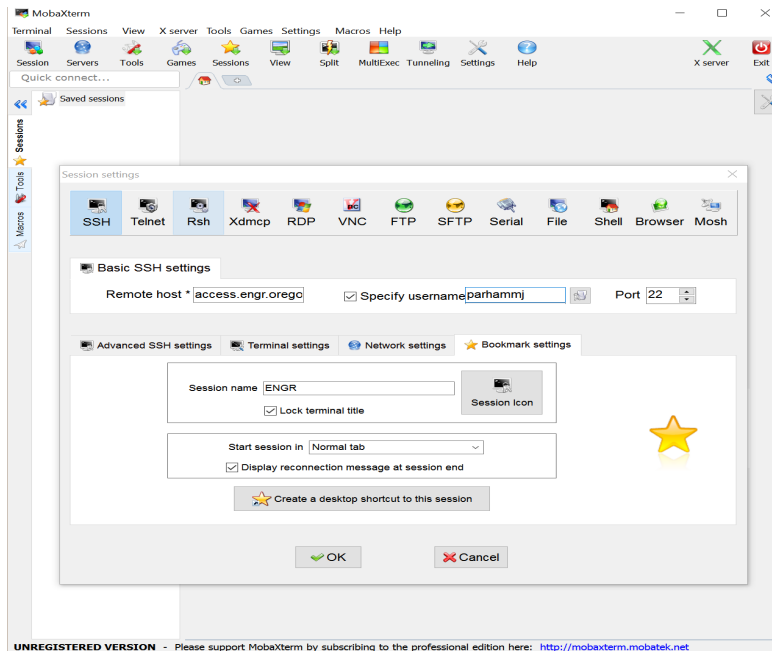
Windows Users:

Download a free Secure Shell (ssh). We will use the MobaXterm this quarter.
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.)

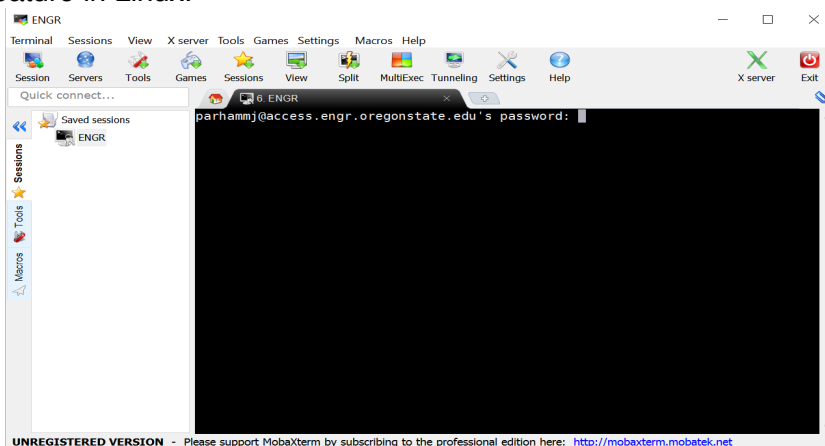
Once you have installed MobaXterm, open MobaXterm. Go to **Sessions -> New Session**, and click on the **SSH icon**.

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.



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.



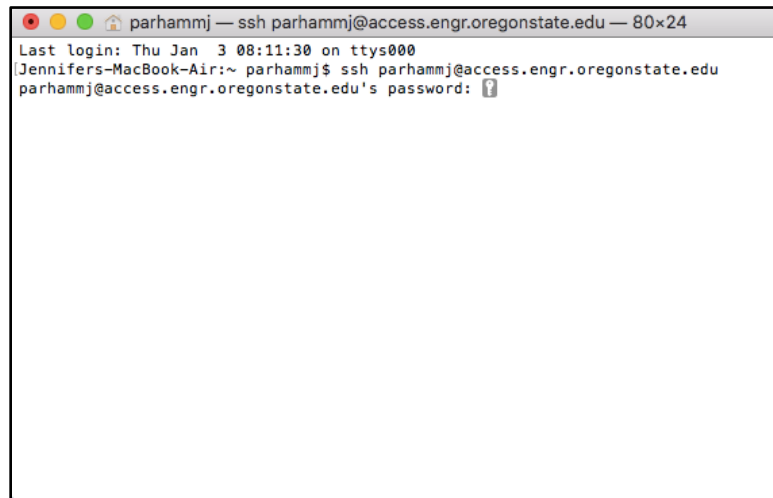
You will need an authentication for DUO Two-factor login. Please read this [information](#) and sign up [here](#). After your first login, you can bypass the need for DUO by setting up SSH Keys using these [instructions](#).

After a successful logon, you should see a prompt that looks something like this:
flip1 ~ 1\$

MacOS or Linux Users:

You have a terminal with ssh built into the OS. Open the terminal by going to the Finder and selecting **Go -> Utilities -> Terminal** (on a Linux system, you might be able to click on the desktop and select Terminal). At the prompt, type **ssh**
<onid>@access.engr.oregonstate.edu .

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.

A screenshot of a terminal window titled "parhammj — ssh parhammj@access.engr.oregonstate.edu — 80x24". The terminal shows the following text: "Last login: Thu Jan 3 08:11:30 on ttys000", "[Jennifers-MacBook-Air:~ parhammj\$ ssh parhammj@access.engr.oregonstate.edu]", and "parhammj@access.engr.oregonstate.edu's password: ". The password field is represented by a single character, indicating that the password is not visible as it is typed.

You will need an authentication for DUO Two-factor login. Please read this [information](#) and sign up [here](#). After your first login, you can bypass the need for DUO by setting up SSH Keys using these [instructions](#).

After a successful logon, you should see a prompt that looks something like this:

flip1 ~ 1\$

If you get “^?” when you try to delete/backspace:

- Go to your **Terminal Preferences** under the **Advanced** tab and enable "Delete Sends Ctrl-H".

(1 pt) B. Practice Linux Commands

- **Listing the contents of a directory/folder.**

At the prompt, type the following commands to look at your files and directories in your home directory. Note the differences on a piece of paper.

\$ 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.

- **Instructions for Unix/Linux commands.**

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.

```
$ man ls
```

Most linux commands also allow you to specify “--help” to get information about usage.

```
$ ls --help
```

- **Find Unix/Linux commands.**

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, to look up UNIX/Linux commands for editing a file, working with a directory, etc.

```
$ apropos editor
```

```
$ apropos directory
```

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
```

- **Directories/Folders.**

Make a directory in your home directory named labs, and change into the labs directory (note what happens to your prompt).

```
$ mkdir labs
```

```
$ cd labs
```

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

```
$ mkdir lab1
```

```
$ cd lab1
```

Find out your present working directory.

```
$ pwd
```

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 ..
```

```
$ pwd
```

```
$ cd ~
```

```
$ pwd
```

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**.

A good rule of thumb is to use **pwd** at any time to determine where you are, in case you forget. Also, use **ls** as often as you need to see a listing of your current directory.

To receive credit for this section, create a Word or text document on your computer to submit at the end of lab. Name this file **lab1_written.doc** (or **lab1_written.txt**). Put your **name, lab section, lab number (1), and date** at the top of the file and create sections for your answers to parts B., E., and F.

Execute the following command to see a visual depiction of all of your files:

```
$ tree -
```

Copy and paste the output from this command into your Word/text file in section B.

(1 pt) C. Write a Program

Use the vim editor to create a C++ file containing your first C++ program.

```
$ vim lab1_worst_advice.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 provided 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.

You start in Command mode until you press 'i' to change to Insert mode. In Insert mode, every character typed is added to the text in the file; pressing the <Esc> (*Escape*) key turns off Insert mode and switches to Command mode.

Basic Vim 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

Your task: Write a starting C++ program that prints out a single line that contains the **worst advice you can think of for a college freshman**. Use the style guideline on the class website for suggestions on how to format your code:

<http://classes.engr.oregonstate.edu/eecs/winter2020/cs161-020/assignments/cs161-style-guidelines.pdf>

```
/* *****  
** Program: lab1_worst_advice.cpp  
** Author: Your Name  
** Date: 01/08/2020  
** Description: Print worst advice for a college freshman.  
** *****/  
  
#include <iostream>  
  
int main() {  
  
    std::cout << "Never go to office hours. Asking for help is a sign of  
weakness." << std::endl;  
  
    return 0;  
}
```

D. Compile and Run Your Program

We have access to two C++ compilers. You can use either one. Today, try out each one and compare the differences. If you see errors or warnings, compare your program to the above example to figure out what needs to be fixed. Ask for help from a TA or a friend or at any time!

After successfully compiling, execute the compiled program.

GNU compiler

```
$ g++ lab1_worst_advice.cpp -o lab1_worst_advice
$ ./lab1_worst_advice
```

Clang compiler

```
$ clang++ lab1_worst_advice.cpp -o lab1_worst_advice
$ ./lab1_worst_advice
```

(3 pts) E. Experiment: Make mistakes and see what happens!

For each experiment, make a change to the .cpp file, recompile (**g++** and **clang++**), and possibly execute the program (**./lab1_worst_advice**) if there are no errors from the compilation. Remember, you cannot fail an experiment!

Experiment 1: What happens when you forget the semicolon at the end of the `cout` statement? Be specific and note the differences between `g++` and `clang++` error messages.

```
std::cout << "Never go to office hours. Asking for help is a sign of
weakness." << std::endl
```

Experiment 2: What happens when you forget to include `iostream`, i.e., remove `#include <iostream>` from the program? Be specific and note the differences between `g++` and `clang++` error messages.

Experiment 3: What happens when you put your advice on a new line? Be specific and note the differences between `g++` and `clang++` error messages.

```
std::cout <<
"Never go to office hours. Asking for help is a sign of weakness."
<< std::endl;
```

Experiment 4: What happens when you put **part of** your advice on a new line? Be specific and note the differences between `g++` and `clang++` error messages.

```
std::cout << "Never go to office hours.
Asking for help is a sign of weakness." << std::endl;
```

Challenge 1: How would you put the first word of your advice on one line and the rest of the advice on another line **in your program** with only one use of `std::cout`? Write the new version of this line of code to achieve it.

Challenge 2: How would you put the first word of your advice on one line and the rest of the advice on another line **in the output to the screen** during execution with only one use of `std::cout`? Write the new version of this line of code to achieve it.

Before moving on, fix any errors you added, so that the .cpp file compiles and runs correctly.

(1 pt) F. Develop a Plan for Assignment 1

Look over assignment #1 on the website. Think about how you will proceed, based on your understanding of the problem and the knowledge you have at this time (feel free to consult **Rao Lesson 3**). Record your answers to the following questions in your Word or text file. You may discuss the assignment with other students, but your answers (and later, your code) should be **written by you alone**.

1. How many inputs will your program request from the user?
2. How will you decide which data types to use for each variable?
3. Will you predict good fortunes, bad fortunes, or a mixture?

(1 pt) G. Upload Written Document (.doc or .txt) and Program (.cpp) to [TEACH](#)

There are two ways to transfer your files from ENGR to TEACH.

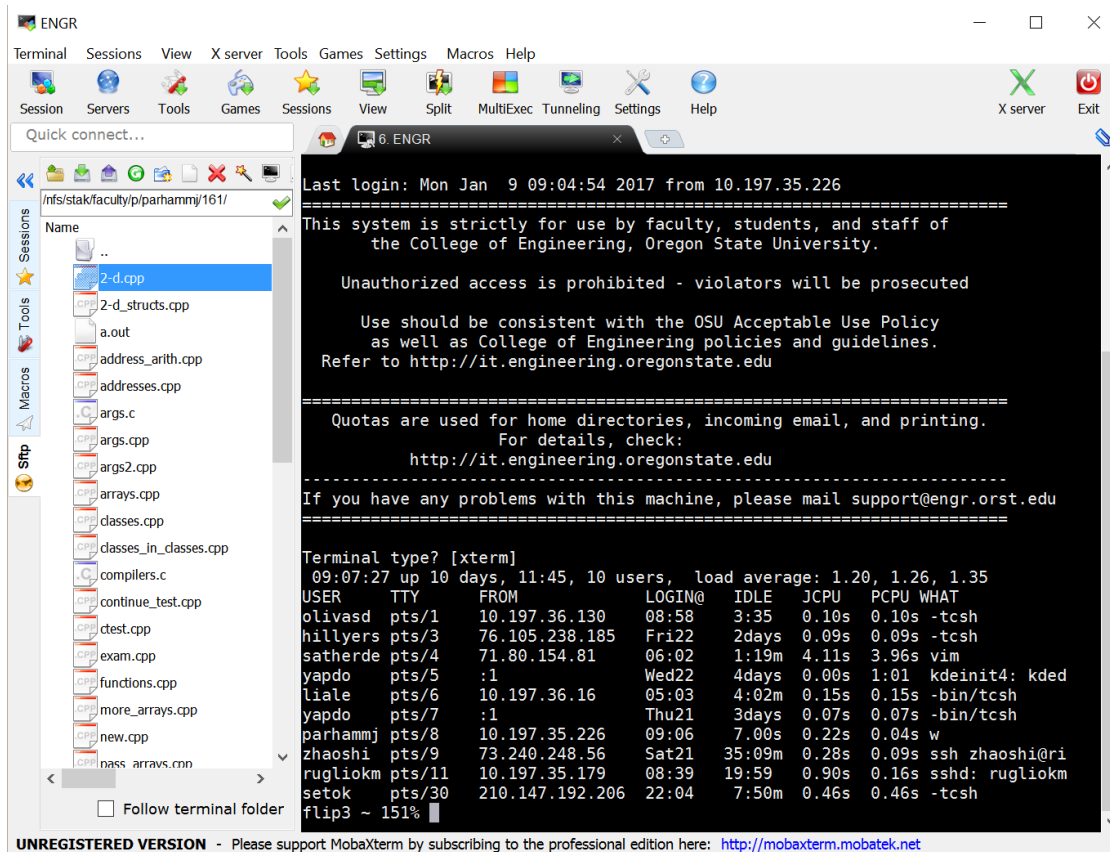
1. Transfer Files from ENGR to local computer

You can transfer files to your own computer, then upload them to 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 hard drive.

Mac OS Users:

- From the command line inside a Terminal window:
\$ scp <onid>@access.engr.oregonstate.edu:<pathname> .
where <pathname> refers to the location of the file you want to transfer. For example:
\$ scp <onid>@access.engr.oregonstate.edu:labs/lab1/lab1_worst_advice.cpp .
Don't forget the final . ! That tells scp where to put the file on your local machine (. means "current directory").
- Using a graphical interface:
Download Cyberduck from <http://oregonstate.edu/helpdocs/osu-applications/offered-apps/ftp>



****NOTE:** You may want to put Cyberduck, MobaXterm, etc. on a flash drive to carry with you. This will enable you to use any computer without needing your laptop all the time.

OR

2. Map Network Drive

You can map a network drive and choose the file from the mapped network drive. 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>

Submit your files to TEACH

1. Connect to TEACH here: <https://teach.engr.oregonstate.edu/teach.php>
2. In the menu on the right side, go to **Class Tools -> Submit Assignment**.
3. Select **CS 161 Lab 1** from the list of assignments and click "SUBMIT NOW".
4. You will upload two files. Individually select your text/Word file with written answers to **sections B., E., and F.** as one file and your .cpp file as the other one.
5. Click the **Submit** button.
6. You are done!

If you finish the lab early, this is a golden chance to start working on Assignment 1 (with TAs nearby to answer questions!).
Remember, the assignment you submit must be your work alone.
