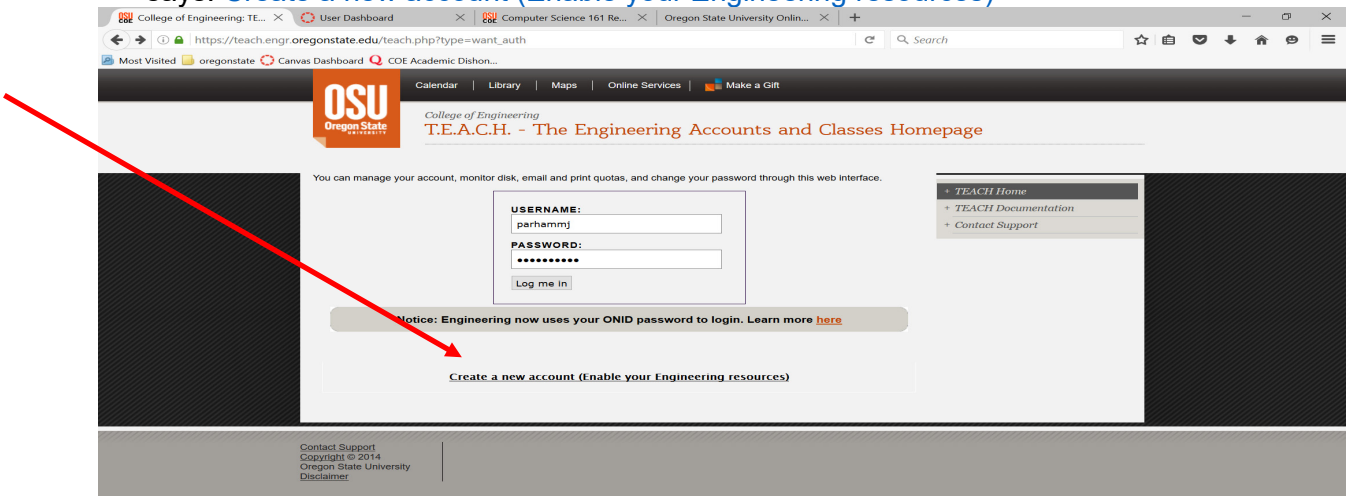
Getting Started on the Engineering Servers

Initial Set Up

1. **If you do not have an ENGR account**, first visit the TEACH page at: https://teach.engr.oregonstate.edu
   You will need to create an account by clicking on the link at the bottom of the login page that says: Create a new account (Enable your Engineering resources)



2. After you have an engineering account, set up your terminal using one of the guides below:
   Windows Instructions
   Mac/Linux Instructions

3. After launching your terminal, at the prompt, type the following commands to look at your files and directories in your home directory.
   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.

Create Directories, Write C++ Program, & Answer Metacognitive Questions

4. Make a directory in your home directory named labs, and change into the labs directory.
   mkdir labs
   cd labs

5. Create a directory in your labs directory named lab1, and change into the lab1 directory.
   mkdir lab1
   cd lab1
6. Make a note of your present working directory.
   pwd

7. 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

8. Now, change into the labs/lab1directory 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**.

9. 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!

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

    A short "cheat sheet" of useful vim commands is available online.
    I have also provided a vim tutorial under **Useful Links** in the Canvas site.

11. Write the infamous “Hello World” program as your first piece of C++ code. Refer to the CS161 style guideline document for formatting information (also available in Canvas on the **Useful Links** page).

```
#include <iostream>

int main() {

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

   return 0;
}
```

12. 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?

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

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

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!";
```

Add a variable and reading input to the program:

13. 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;
}
```

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

15. 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?

Transfer Files

You can transfer the file to your own computer, then upload it on TEACH. Return to the appropriate document for details:

Windows Instructions
Mac/Linux Instructions