Introductory Lab

The purpose of this assignment is to introduce you to the basics of writing, running, and submitting programs in this class. You can work on this assignment from the lab computers in the Joyce Cummings Center—in fact, you may do this for all assignments this semester, but we recommend taking this opportunity to learn how to work remotely, so that you can comfortably work from your own computer.

Important: Do **not** copy and paste the commands we give you. Type them by hand. This goes for the whole semester!
If you copy and paste, you won't actually learn the commands, and that will cost you lots of time later. Learn them now. Learn them by typing, observing, and thinking.
Copy and paste is evil!
And as always, don't be afraid to ask questions.

You can work remotely by accessing the CS department's Linux computers—we call these computers *servers* because they remotely provide a service to us, the clients.

We will learn the basic ideas and skills of using a Linux system. Linux is an *operating system*, just like macOS or Windows. We commonly operate a Linux system using the text-based *terminal* mode. But first, we must log in to the CS department's server.

0.1 Remotely Logging In

We will access the department servers by using `ssh`, a popular protocol for remotely operating computers. To use `ssh`, you'll first need to open a terminal:

- **On Mac/Linux**: Open your native terminal program. On Mac, that's done using `Applications -> Utilities -> Terminal` in Finder, or by pressing the `command` and space bar on your keyboard at the same time and then typing in `terminal`.
- **On Windows**: You'll need to download the program MobaXterm from `http://mobaxterm.mobatek.net/`. Once installed, open it up. You may want to go to `Settings/Configuration -> terminal` tab, and uncheck "Backspace sends ^H".

When you get a terminal, type

`ssh` *`username`*`@homework.cs.tufts.edu`

where "username" is your CS department username. You will be prompted for your password. Once logged in, you will now be operating on the department server!

You are operating on the department server using the *command line*, a text-based way of navigating a computer. In reality, you are just running another computer program called the Linux *shell*.

To confirm you successfully logged in, try typing `hostname` into the command line and hitting enter. You should get back the name of the department machine you are operating on, which will look something like `vm-hwXX`, where `XX` is replaced by numbers. **NOTE:** if you see a name that looks different from this, you are probably not logged in correctly—it could just be the name of your own computer! Re-follow the above steps until you see a hostname that looks like the above.

Make sure you are able to log in following these instructions. You will need to do this a lot this semester! If you have any questions or concerns about remotely logging in, ask them now.

0.2 Using the Command Line

You have successfully accessed the department server! Now, let's try running some commands.

⇒ Type in the following commands one at a time and see what they do:
```
hostname
who
date
cal
cal 2020
cal 1 2020
cal 1 20
```

⇒ Use the `cal` command to find out what day of the week you were born.

The things you just typed are called commands. These are names of programs. Nowadays, people refer to programs by the term "apps." We shall refer to them as commands, programs, or tools.

- **command** — the name of a program, like `cal`
- **arguments** — The extra information you give to a command to specify what it should do. For example, the number `2020` tells the `cal` program to print the calendar for the year 2020. If you do not specify arguments, the command will perform its default action.

The Manual All commands are explained in the online help system, called the *manual*. To learn about the `cal` tool, for example, use the `man` tool:

⇒ `man cal`

This will display the help page for the `cal` tool. These help pages can be terse but have all information about the command. You can exit out of a manual by typing `q`.

These days, this information is also available on the internet. You can search there. The system `man` information, however, is the definitive source of information for your system.

0.3 Files and Directories

For this class, all programs you write will be stored in folders in your account on the server. We will now look at the basic commands for managing files and folders.

In Linux, a folder is called a *directory*. When using the command line, you are always located in a single directory within your computer's file system. The directory you are located in is called your *working directory*. When you first log in, you are placed in your *home directory*.

⇒ Type "`pwd`" and hit enter (this stands for *"print working directory"*). This will show you the name of the directory you are currently located in—your working directory! It should look like `/h/USERNAME`, where `USERNAME` is replaced by your CS login name. This directory is also your home directory, and it is the personal space where you will start every time you log in.

⇒ Type "`ls`". This will list all the folders and files in the directory that you are in. If you have never logged in to the department servers before, you probably don't see anything when you type `ls`.

⇒ Let's make a new directory. Type "`mkdir folder1`" and hit enter. This will—you guessed it—make a new directory called `folder1`. Now try typing "`ls`". You should see the new directory.

⇒ Let's move into our new folder. Type "`cd folder1`". The `cd` command changes into the directory that you give it. Type "`pwd`". Your working directory should now be `/h/USERNAME/folder1`. Changing directories is analogous to double clicking on a folder on your Mac or Windows computer.

⇒ Now type "`cd ..`" and hit enter. The `..` moves you up a directory. Now when you type `pwd` you should be back in your home directory.

⇒ Type "`mv folder1 folder2`" and hit enter. Now type `ls`; you should see `folder2`. The `mv` command moves the first file/directory that you give it to the second file/directory. In this case, we have just renamed `folder1` to `folder2`.

⇒ Now let's delete the directory we made. Type `rm -r folder2`. The `rm` command removes a directory or file. The `-r`, which is known as a "flag", tells `rm` that we are removing a directory. Type `ls`; the directory should now be gone. **WARNING:** The `rm` command *permanently* removes a file. There is no temporary trash bin! Use it carefully.

Important:
You will be navigating the command line all semester long, so it is important to know these commands. You can use this as a reference:

`pwd` stands for *print working directory*, and it will always tell you where you are.

`cd` *dirname* will take you to the directory named *dirname* (if it exists and is accessible to you).

`cd ..` takes you one directory level up, e.g., if you are in /h/milod/folder1/ it will take you to /h/milod/.

`cd`, when called with no arguments, will take you to your home directory, so you can use it if you get lost.

`cd ~` will also take you to your home directory, because `~` is a shorthand for "my home directory."

`ls` will list all the files and folders in your current directory.

`mv name1 name2` moves the file/directory called `name1` to `name2`.

`rm [-r] name` will permanently remove the file/directory called `name`. When `name` is a directory, the `-r` flag is required.

0.4 Compiling and Running Programs

Now that you know your way around the command line, you are ready to compile and run your first program. We will provide you with this first program.

Make sure you are in your home directory (type `cd`). Make a directory here called "cs15" (`mkdir cs15`), and `cd` into it (`cd cs15`). You should put all of your work for this course inside this directory. Now, make a directory called "lab0" and `cd` into it. Use `pwd` to make sure you are in the right place—you should be in /h/USERNAME/cs15/lab0.

Now, copy the first program we have provided for you by typing
`cp /comp/15/files/lab_intro/welcome.cpp .` and hitting enter. Note that *you must also type the dot at the end*! The `cp` command copies files and we have given it the path to the program file that we want to copy, as well as the dot symbol `.`, which tells `cp` to copy the file to the current working directory. You can use `ls` to make sure you have successfully copied the file.

The filename ends with `.cpp`, indicating that it is a C++ program. In a moment, we'll learn how to view and edit this program. For now, let's just compile and run it. Compile the program by typing the following command:

```
clang++ -Wall -Wextra welcome.cpp
```

This will compile your *source code* and generate an *executable program* called `a.out`. You should be able to see `a.out` when you type `ls`.

Run your program by typing "`./a.out`" and hitting enter. It should print a message welcoming you to CS 15. Did it work? If so, celebrate. If not, try to figure out what happened. We'll help!

0.5 Writing Programs with VS Code

There are many popular *editors* for writing/editing programs: emacs, vim, Sublime, VS Code.... If you already have an editor that you're comfortable with, then you're good to go! Make sure that you can use your editor to remotely access the files on the CS department's servers.

We recommend using *Visual Studio Code* in CS15. You will download and install VS Code on your personal computer, and use it to remotely write and edit programs that are on the department servers in Halligan Hall.

You can find instructions for setting up VS Code on our course website: https://www.cs.tufts.edu/comp/15/reference/system_setup/vscode_setup.pdf

These instructions also detail how to install the SFTP extension for VS Code and use it to remotely sync with your `cs15` folder.

Once you have setup VS Code and SFTP, you should see all of your directories from the CS server listed on the left hand side of VS Code (at a minimum, you should see your `cs15` folder containing `welcome.cpp`). If you do not, try to re-follow the VS Code setup instructions. If you're still having trouble, ask us for help. It's crucial that you are properly synced!

Now, from VS Code, open and edit the `welcome.cpp` program. Change the comment at the top of the file so that it includes your name. Then, below the line where the welcome message is printed, add a second line that prints the text "NAME's first program.", replacing "NAME" with your name, of course. Save your changes.

Now, switch back to terminal. You will always use VS Code to write/edit programs, then actually run those programs from the command line. Make sure you are still logged into the department servers—you can do this by typing `hostname` and checking that you get the same output as before. Use `ls` to make sure you are in the same directory as your `welcome.cpp` program. If not, navigate to it. If you are, recompile the program:

```
clang++ -Wall -Wextra welcome.cpp
```

Every time you edit a program, you have to recompile it before running it again. Finally, run the program by typing "./a.out" and hitting enter. You should see an output that looks as follows:

```
Welome to CS 15!
NAME's first program.
```

Does the new text with your name appear in the output? If so, congratulations! You have completed your first program. If not, see if you can figure out what went wrong. If you are getting an error message, try to figure out what it is telling you. If you are still having trouble, ask us for help.

0.6 Submitting your work

In CS 15, all labs and HW assignments will be turned in on Gradescope, via your browser. Whenever you submit, do not submit a folder. Just submit the files inside the folder you wish to submit. Make sure that your local versions of your files match what is on Halligan. Pay close attention to the submission section of each assignment to know which files you will need to submit.

You will need to submit the following files:

welcome.cpp

You must submit them using Gradescope to the assignment `lab_intro`. Submit these files directly, do not try to submit them in a folder.

Congratulations, you're a C++ programmer now!