15-122: Principles of Imperative Computation — Spring 2020

Lab 1: Setup — Tuesday January 14th

This first lab is UNGRADED.

As you start this lab (and any future lab), please take care to read an *entire* part before doing anything the part asks you to do. There are often important caveats or secondary instructions that we have to give *after* the initial instruction in order for them to make sense.

Getting started

The first thing you will do is to check out Diderot, our online Q&A platform.

Go to `https://www.diderot.one/course/26`, and you should find our course.

Next, you'll start getting acquainted with Linux:

If you are on a classroom computer, and generally on a Windows machine, you can access a Terminal window by opening PuTTY. You can do that by hitting the Start menu, and typing "PuTTY" in the search bar. If nothing shows up, download it from this link: `https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe`. Next, follow these steps:

(a) In the Hostname bar, enter "YOUR_ANDREW_ID@unix.qatar.cmu.edu"

(b) Keep the port number as 22.

(c) Save these settings by entering a name (say, 'CMU') in the box right under "Saved Sessions." Afterward, hit "Save" on the right hand side.

(d) Next, click on the "Open" button.

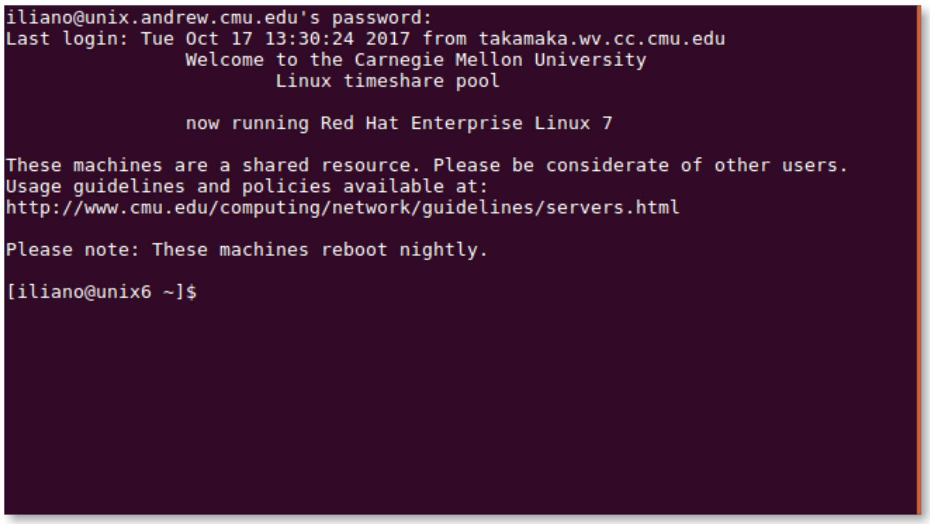
(e) You will now see a new Terminal window open, and will ask you for your password.

(f) Enter your password; you will not see any characters being entered (for security/privacy!) so don't freak out! Once you type your password, hit "Enter".

If you are on your own laptop, you need to log in to one of the andrew Linux machines. How to do so depends on what OS your laptop is running (Windows, Mac, or Linux). Go to Diderot and follow the instructions in the post "`Laptop Setup for <OS>`" where `<OS>` is the OS you are running on your laptop.

If you are on a cluster machine but plan to use your laptop for assignments in this course, you will want to do the laptop setup on your own later.

Once you are done, you will see a terminal window that looks something like this:



```
iliano@unix.andrew.cmu.edu's password:
Last login: Tue Oct 17 13:30:24 2017 from takamaka.wv.cc.cmu.edu
                Welcome to the Carnegie Mellon University
                        Linux timeshare pool

                now running Red Hat Enterprise Linux 7

These machines are a shared resource. Please be considerate of other users.
Usage guidelines and policies available at:
http://www.cmu.edu/computing/network/guidelines/servers.html

Please note: These machines reboot nightly.

[iliano@unix6 ~]$
```

(The messages will look different for you.)

Navigating your account in Linux

Linux is an operating system, just like Windows or Mac OS X. If you want to know a little more about what's going on here, look for the Diderot post titled "What is Linux?".

In this lab, rather than using a graphical interface, you will use the terminal to enter commands directly to the OS.

(2.a) At the prompt, list the files in your home directory[1] by typing

```
% ls
```

We often use the `%` or `$` character at the beginning of a line to indicate that it's something you're supposed to enter at the prompt. Don't actually type it in! Just type "`ls`" and press Enter, don't type "`% ls`" and then press Enter.

(2.b) You should see the directory `private` as one of the entries in your home directory. Move to this directory using the `cd` command, which lets you change the directory your terminal is working in.

```
% cd private
```

ACADEMIC INTEGRITY NOTE: You should store your program files and other class solutions inside the `private` directory (or a subdirectory inside this directory) since this directory is automatically set to prevent electronic access by other users. Remember that you should protect your work from being accessed by other students as part of the academic integrity policy for this course.

(2.c) Once you `cd` into the `private` directory, make a new directory named `15122` using the `mkdir` command:

```
% mkdir 15122
```

Now go into this directory using `cd` again:

```
% cd 15122
```

Setting up your Linux Account

Now that you know how to navigate in the Linux machine, it needs to be set up so that you have syntax highlighting and can use the course tools.

(3.a) At the prompt, enter the command

```
% wget https://web2.qatar.cmu.edu/~srazak/courses/15122-s20/lab/15122-setup.sh
```

and execute it

```
  % bash 15122-setup.sh
- files modified: ~/.bashrc, ~/emacs, ~/.vimrc (older versions are backed up)
- your default shell will now be bash
You do *not* need to run this script again
```

If you run into difficulties, ask a TA. You can alternatively look at the instructions at `http://c0.typesafety.net/tutorial/Setting-up-your-environment.html` (but you shouldn't need to).

[1]You're probably familiar with "folders" if you've used Mac or Windows. "Directory" is the Linux name for folders.

Completing a Programming Assignment

In this section, we'll walk through the steps of downloading, completing, and submitting a programming assignment. You'll do this for each assignment, so come back here if you get confused.

(4.a) Download the handout for this sample assignment by going to Autolab (https://autolab.andrew.cmu.edu), clicking on "Sample Assignment (UNGRADED)" and then on "Download handout".

If you are on your own laptop, you will need to transfer the file `sample-handout.tgz` to andrew so that you can work on it on that system.

If you are on a Mac or a Linux laptop, open a new terminal and `cd` to the directory where your browser saved `sample-handout.tgz` (it's probably "Downloads").
We'll use the `scp` command to copy the file from your laptop to the Andrew servers. `scp` is for securely copying files over a network. You tell it where the file is now (the "source") and then where you want the file to go (the "destination"). Run the command

$$\texttt{\% scp}\quad \underbrace{\texttt{sample-handout.tgz}}_{\text{source}}\quad \underbrace{\underbrace{\texttt{your\_id@unix.qatar.cmu.edu}}_{\text{the computer we're sending the file to}}\texttt{:}\underbrace{\texttt{private/15122/}}_{\text{where to put the file}}}_{\text{destination}}$$

It will ask for your andrew password. Once you enter it, it will transfer the file to andrew. Go back to the terminal where you're `ssh`'d to andrew and in the `private/15122` directory. Run `ls` to see the file we just transferred:

```
% ls
sample-handout.tgz
```

If you are on a Windows laptop or a cluster computer, you would need another program to be able to upload the files that you downloaded to the andrew server. One such program is called FileZilla. You might not have to download it; hit the Start menu and type "FileZilla" and open it up. Next, follow the instructions on the link below to set it up: `http://alexcappiello.com/15122-misc/winssh.shtml#filezilla`
Afterward, navigate to your lab folder you were working on (Should be private -> 15122) and upload (click and drag) the file you just downloaded.
Go back to the terminal where you're `ssh`'d to andrew and in the `private/15122` directory. Run `ls` to see the new file:

```
% ls
sample-handout.tgz
```

(4.b) `sample-handout.tgz` is a compressed archive (like a `.zip` file) consisting of several files. You can retrieve them with the following command [2]:

```
% tar xfzv sample-handout.tgz
sample-handout/
sample-handout/factorial.c0
sample-handout/favorite_number.c0
sample-handout/README.txt
```

[2] If this command fails, try `tar xvf sample-handout.tgz`.

You've now created the directory `sample-handout` containing the three files `README.txt`, `factorial.c0`, and `favorite_number.c0`. Go to this directory:

```
% cd sample-handout
```

(4.c) Verify you are in the `sample-handout` directory by entering the command `pwd` to get the present working directory and by entering the command `ls` to see the files in that directory. You should see something like this with your andrew id instead of `<your_id>`:

```
% pwd
/afs/qatar.cmu.edu/<usr_number>/<your_id>/private/15122/sample-handout
% ls
factorial.c0  favorite_number.c0  README.txt
```

Editing your program

You can use any editor you wish to write and edit your programs, but we highly recommend you try out `vim` and `emacs` since these editors can do much more than just help you edit your code (as you will see). For this lab, **try both of them** by following the instructions below. Later, use the one you like best[3].

(4.d) Open the file `factorial.c0` that you downloaded from the previous part of the lab:

VIM:
```
% vim factorial.c0
```
EMACS:
```
% emacs factorial.c0
```

If a file doesn't exist, the editor will start with a new empty file. You should see the editor start in the Terminal window, and you should see a program that looks like it computes factorial. The program is written in C0, the language we'll be using to start the semester.

(4.e) Edit the program and add your name and section letter at the appropriate locations. Use the instructions below for the editor you're using.

VIM: This editor has two modes, *insert mode* where you can insert text, and *command mode* where you can enter commands. It starts in command mode so you can't edit immediately. Use the arrow keys to move around the file. While in command mode, if you press "`i`", the editor changes you to insert mode, allowing you to type text. Go to insert mode and add your name and section letter. Press the Escape (ESC) key while in insert mode to return to command mode.

EMACS: You can just start typing and editing without hitting special keys. You can use the arrow keys to navigate around the file to insert code. There are many shortcuts and built-in features to emacs but you don't need them right now. In the file, insert your name and your section letter in the appropriate comments in your program.

[3]See the course website for some links to learn more about using these editors. They both are capable of a lot more than we describe here.

(4.f) Save your changes and exit the editor.

VIM: Make sure you're in command mode by pressing ESC. Then, you can save your work and exit vim by entering the sequence "`:wq`" followed by pressing Enter. (If you have unsaved changes you would like to discard, you'll have to enter the sequence "`:q!`" followed by Enter. You can also save without exiting by entering the sequence "`:w`" followed by Enter.)

EMACS: Once you're ready to save, press Ctrl-x (the Control key and the "`x`" key at the same time) followed by Ctrl-s. You can exit by pressing Ctrl-x followed by Ctrl-c. If you have not saved before exiting, Emacs will ask you whether you want to save your file (since you changed it) — press "`y`" for yes. (Press "`n`" instead if you don't want to save your changes).

(4.g) Try out your new editing skills on the file `favorite_number.c0`. You should see where the file tells you to add a line that returns your favorite number, like this:

```
int my_favorite_number() {
    /* add a line below that returns your favorite number */
    return 17; // this is *my* favorite number. Choose your own.
}
```

Running a C0 Program

You can execute a C0 program by either compiling it into executable code or loading it into an interpreter. You invoke the C0 compiler (`cc0`) and interpreter (`coin`) from the command line.

The *compiler* translates your program into a lower level (machine) version of the code that can be executed by your computer. It first checks for syntax errors and will abort with an error message if it finds a syntax error.

(4.h) Compile your code using the `cc0` compiler:

```
% cc0 -d factorial.c0
```

This runs the compiler with debug mode on (`-d`). During execution, the debug mode checks all the code annotations starting with `//@`. You will learn about them in class.

If there are no syntax errors, the `cc0` compiler returns to the command prompt without saying anything else. Running `ls` will show you a new file named `a.out`, which is the executable version of your program. If you have syntax errors during compilation, go back into the file with an editor and correct them.

(4.i) Run the program:

```
% ./a.out
```

The first dot says to look in the current directory and run the `a.out` executable file. This will cause the `main()` function in your program to launch, which prints the values of 0! through 9! in the terminal window, one per line.

Alternatively, you can use the C0 interpreter to execute your program. An *interpreter* checks a program for syntax errors and runs it step by step. This is a good way to interact with your program in real time to test it.

(4.j) Run your program in the `coin` interpreter, starting it with `coin -d  factorial.c0` and entering in the C0 statements as shown below.

```
% coin -d factorial.c0
C0 interpreter (coin)
Type ‘#help’ for help or ‘#quit’ to exit.
--> factorial(2);
--> factorial(3);
```

Exit the interpreter by entering `#quit`.

Submitting Programming Assignments

(4.k) You will submit your *programming* assignments using Autolab, the site where you downloaded the handout from. Let's see how that works on the files `factorial.c0` and `favorite_number.c0`.

We will always submit compressed files. You do so by typing the following at the Unix prompt:

```
% tar cfzv handin.tgz factorial.c0 favorite_number.c0
```

The above command puts your files `factorial.c0` and `favorite_number.c0` into a compressed archive file named `handin.tgz`.

If you are on your own laptop, you need to move the `handin.tgz` file from the andrew server back to your own laptop.

If your laptop is running Linux or Mac, open a new terminal. We'll use the `scp` command from earlier.

$$\%\ \texttt{scp}\ \underbrace{\underbrace{\texttt{your\_id@unix.qatar.cmu.edu}}_{\text{the computer the file is on now}}:\underbrace{\texttt{private/15122/sample-handout/handin.tgz}}_{\text{where the file is now}}}_{\text{source}}\ \underbrace{\texttt{.}}_{\text{dest}}$$

The `.` at the end means "here" — it tells `scp` to put the file in the current directory your terminal is in. If you didn't `cd` anywhere else, it's the home directory on your laptop.

If you are on a Windows laptop or a cluster computer, use FileZilla to download the file to the Windows computer. Navigate to your lab folder you were working on (Should be private -> 15122 -> sample-handout) and download (click and drag) the tar file you just created.

Next, point your browser to Autolab, go to the sample assignment's entry and upload the newly created file `handin.tgz` using the "Submit File" button.

Check that you submitted the right file by clicking on the "view source" icon. **Always check your submissions!**

There's also an easier way to submit from an andrew computer. You can use the `handin` command to hand in your work. The file `README.txt` in each handout will tell you how to submit using `handin`. Look at the `README` for this sample homework to see what to do (you can view the file by opening it in an editor, like Vim or Emacs).

Written Assignments

(5.a) You will submit your *written* assignments using Gradescope. But first we need to register at https://gradescope.com **using YOUR ANDREW EMAIL** and entry code **9V64DR**.

Written homeworks are posted on Diderot. In a browser, go to Diderot, and click on "`Written 0 (UNGRADED)`" under "Written Homeworks". The printer icon in the top right corner will give you a version you can save as a PDF. Save a copy as `written-test.pdf`. You will write just your name and section number and then submit.

You can do the writing in two ways:

(a) By entering annotations in `written-test.pdf` and then saving. The easiest way is to do so online by using `pdfescape.com` in your browser. Alternatively, several PDF viewers support annotations, including `preview` on Mac, `iAnnotate` on iOS and Android, and `Acrobat Pro` on pretty much anything — `Acrobat Pro` is installed in all non-CS cluster machines.

(b) By printing `written-test.pdf`, writing your solution by hand, and then scanning it back to PDF. *This is pretty laborious: you will want to get the first option to work for you.*

Now that you have a "solved" version of `written-test.pdf`, go to Gradescope and upload your solution file. ***Always check your submission to be sure it looks correct!*** If you didn't manage to find a PDF editor that works for you, simply submit the blank writeup for now.