

This first lab is UNGRADED.

**Important Info:**

Before starting 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 enroll in Piazza, our online Q&A platform.

Go to <https://piazza.com/cmu/spring2019/15122> and join the class.

Next, you will start getting acquainted with Linux. How to do so depends on whether you are doing this lab on one of the GHC cluster computers or on your laptop. *You need to do only one of the following two tasks right now.*

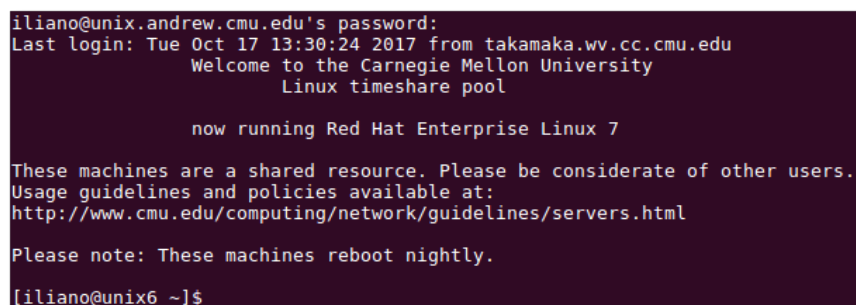
- (1.a) **If you are on a GHC cluster computer**, log in and then open up a Terminal window to access the Linux command prompt.

On the Linux machines in the computer labs, you can access a Terminal window using the menu sequence: Applications → System Tools → Terminal. Your terminal window will give you a prompt and you will be in your home directory in your Andrew account.

- (1.b) **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 Piazza and follow the instructions in the post “Laptop Setup for <OS>” where <OS> is the OS you are running on your laptop.

**Hint:** If you are on a cluster machine but plan to use your laptop for assignments in this course, you may want to do task (1.b) on your own later.

Once you are done, you will see a terminal window that looks 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

Unlike with typical graphical interfaces for operating systems, in Unix you are entering commands directly to the OS, which enables you to change hundreds of options to finely control what you're doing.

- (2.a) At the prompt, list the files in your home directory 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 (`cd` stands for "change directory"):

```
% 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) Since you will write a number of programs for this course, it pays to make a subdirectory inside the `private` directory. 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
```

Let's create a directory for this lab, 'lab01', and go to it:

```
% mkdir lab01
```

```
% cd lab01
```

- (2.d) Download the handout for this lab by firing up a browser (e.g., Firefox), pointing it to Autolab (<https://autolab.andrew.cmu.edu>), going to "Lab 1: setup" (that's presumably where you got this writeup from) and clicking on "Download handout". There is also a copy of the handout on Piazza.

*If you are on a GHC cluster computer*, this will save the handout as the file `handout-01.tgz` in the 'Downloads' directory under the directory Terminal started at (that's your *home directory*). This is a compressed archive consisting of several files. You can retrieve them with the following command:

```
% tar xfv ~/Downloads/handout-01.tgz
handout/
handout/factorial.c0
```

*If you are on your own laptop*, you will need to transfer the file `handout-01.tgz` on andrew so that you can access it through your terminal. *If you are on a Mac or a Linux laptop*, open a terminal, `cd` to the directory where your browser saved `handout-01.tgz` and run the command

```
% scp handout-01.tgz <your_id>@unix.andrew.cmu.edu:private/15122/lab01
```

It will ask for your andrew password. Once you enter it, it will transfer the file to andrew. You can check by running `ls` on your andrew terminal: it should be there! You can uncompress this file by running

```
% tar xfv handout-01.tgz
handout/
handout/factorial.c0
```

If this command fails, try `tar xfv handout-01.tgz` (without the `z`).

*If you are on Windows*, you can transfer the `handout-01.tgz` interactively using one of the icons on MobaXterm. Ask a TA if you can't figure it out. Then, proceed as above.

This creates a directory called `handout` containing the file `factorial.c0`. Go to this directory:

```
% cd handout
```

- (2.e) Verify you are in the `lab01/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/andrew.cmu.edu/<usr_number>/<your_id>/private/15122/lab01/handout
% ls
factorial.c0
```

## Setting up your Linux Machine

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 C0 interpreter (`coin`).

- (3.a) Next, we will set up your environment to conveniently run the programs used in this course. At the prompt, enter the command

```
% /afs/andrew/course/15/122/bin/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 instruction at <http://c0.typesafety.net/tutorial/Setting-up-your-environment.html> (but you shouldn't need to).

## Editing your program

You can use any editor you wish to write and edit your programs, but we highly recommend you try out `emacs` and `vim` 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.

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

**EMACS:**

```
% emacs factorial.c0
```

**VIM:**

```
% vim 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 some program that looks like it computes factorial. The program is written in C0, the language we'll be using to start the semester.

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

**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.

**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. In the file, insert your name and your section letter in the appropriate comments in your program. Press the Escape (ESC) key while in insert mode to return to command mode.

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

**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. (You could press "n" instead if you don't want to save your changes).

**VIM:** *You can only save and/or exit while in command mode.* Save your work and exit the editor by entering the sequence `:wq` followed by pressing Enter. (You can exit without saving by entering the sequence `:q` followed by 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.)

If you'd like to learn more about emacs editing commands in your own time, try this tutorial [http://cs.cmu.edu/~15122/misc/emacs\\_quickref.pdf](http://cs.cmu.edu/~15122/misc/emacs_quickref.pdf).

To learn more about vim, you can try out `vimtutor` in your own time. Simply type

```
% vimtutor
```

at the command prompt in the Terminal window to get started.

## 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.

(5.a) 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.

(5.b) 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.

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

```
% coin -d factorial.c0
C0 interpreter (coin)
Type '#help' for help or '#quit' to exit.
--> factorial(2);
--> factorial(3);
--> factorial(4);
--> factorial(10);
--> factorial(17);
```

Some of the outputs `coin` gives should strike you as odd. We'll learn about what's going on in class.

(5.d) Factorial  $n!$  is only defined on non-negative numbers. Try to compute a non-existent factorial:

```
--> factorial(-1);
```

You should see an annotation failure. This is because our factorial function starts with the requirement: `//@requires n >= 0;`. Since we called this function with a value for  $n$  that does not satisfy this requirement, we get an annotation failure since it doesn't make sense to run this function with  $n = -1$ .

(5.e) Exit the interpreter by entering `#quit`. Start it again, this time without the `-d` flag by typing `coin factorial.c0` at the prompt. Now, run `factorial(-1);` again. What do you observe?

## Submitting Assignments

In this class you will be submitting your assignments using two tools that may be new to you. This activity has the purpose to make sure you know what you are doing.

### Written Assignments

- (6.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 **M3PWPJ**.

Because Gradescope doesn't support posting homeworks, we do so on Autolab. In a browser, go to Autolab, click on "Written 0 (UNGRADED)" and then on "View writeup". 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](https://pdfescape.com) in your browser. Alternatively, several (but not all) PDF viewers support annotations. Examples are `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.*

Unless you are near a scanner, you will use the first option.

Now that you have a "solved" version of `written-test.pdf`, point your browser to Gradescope and upload your solution file. That's it! If you didn't manage to find a PDF editor that works for you, simply submit the blank writeup for now.

### Programming Assignments

- (6.b) You will submit your *programming* assignments (and some lab solutions) using Autolab, the site where you downloaded the handout from. Let's see how that works on the file `factorial.c0` where you earlier wrote your name and section number.

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

```
% tar cfzv testing.tgz factorial.c0
```

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

(There is also a way to submit from the prompt. You'll see that later on.)

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

## Quizzes

Throughout the semester, we will occasionally have a quiz in class or recitation — *don't be afraid of quizzes: they are the best way to get participation points!* Let's make sure you are all set.

(7.a) Point your browser to the quiz page (<http://cs.cmu.edu/~15122/quiz.shtml>) and click on the **Test** button. Several things could happen:

- (a) You get an error message. Go back to the quiz page and follow the instructions under “Common misconfigurations”. Then try again.
- (b) You end up in your personal Google account. Click on the top right icon. If your andrew email address is listed, click on it and continue with step (c). If you don't see your andrew email address, click on “Add account” to enter it and then continue with step (c).
- (c) You get to a Google sign-in page. Authenticate with your **andrew email address**. Then proceed to step (d).
- (d) You are redirected to CMU's web login page. Authenticate and proceed to step (e).
- (e) You see a Google form entitled “Testing your configuration” — good! Click on the radio button and then **submit**. You are ready to take quizzes!

If you get stuck at any point, ask a TA to help you.

At this point you are done with the lab, and you are free to go.