

15-122: Principles of Imperative Computation (Qatar) — Spring 2025

Setup Lab

Complete by 9pm on Wednesday 8th January, 2025

This document guides you through configuring your computer for 15-122 and setting up the technology used in this class.

Getting started

- ▶ The first thing you will do is join our Discord, our online Q&A platform.

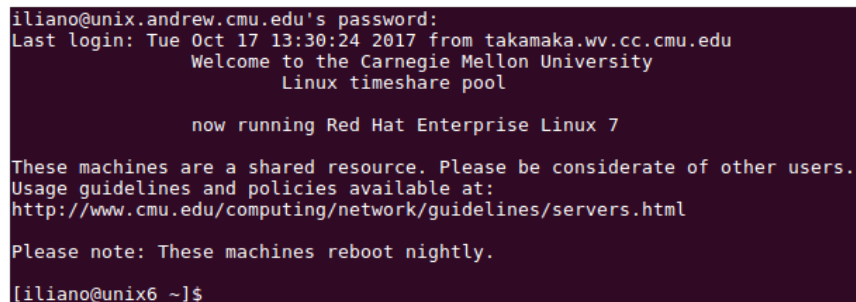
Click on the Discord signup link <https://discord.gg/ewqMMKdT>.

- ▶ Next, you'll start getting acquainted with Linux.

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 [Discord](#) and follow the instructions in the post “Laptop Setup for <OS>” where <OS> is the OS you are running on your laptop.

You should do the same from a cluster computer on campus: if your laptop fails, you will want to be set up to continue your work from a cluster machine.

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.

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

- ▶ At the prompt, **list** the files in your home directory¹ 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, not "`% ls`".

- ▶ 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.

- ▶ 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.

- ▶ At the prompt, enter the command

```
% /afs/andrew/course/15/122q/bin/qatar-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 https://bitbucket.org/c0-lang/docs/wiki/Setting_up_your_environment (but you shouldn't need to).

¹You'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.

- ▶ The first thing you will do is check you are enrolled in [Autolab](#), the platform where will submit your programming assignments. Go to [Autolab](#) and find our course among your list of courses. After this, you will be primarily using the command `autolab122` to interact with [Autolab](#). You can also use the [Autolab](#) web site (<https://autolab.andrew.cmu.edu>) if you prefer.
- ▶ To begin with, you need to do a one-time setup for `autolab122`. Simply run

```
% autolab122 setup
```

and follow the instructions. You will not need to do this again. Here's a sample interaction.

```
astanesc@unix4 ~ $ autolab122 setup

I affirm that, by using this product, I have complied and always will comply with my
courses' academic integrity policies as defined by the respective syllabi [Y/n].
Y
Initiating authorization...

Please visit https://autolab.andrew.cmu.edu/activate and enter the code: DKMtmQ

Waiting for user authorization ...
Received authorization!

User setup complete.
```

- ▶ Make sure you are in the `private/15122` directory by entering the command `pwd` to get the present working 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
```

If you ever get lost, you can go there by entering

```
% cd
% cd private/15122
```

- ▶ You can list the currently released assignments by running

```
% autolab122 hw
sample (Sample Assignment (UNGRADED))
```

You will see the assignment called `sample` — the description in parentheses is for your information only — and possibly other material we are distributing through [Autolab](#). You will be using this name (`sample`) when interacting with Autolab about this assignment.

- Download the starter code for assignment sample with the following command:

```
% autolab122 download sample
Querying assessment 'sample' of course '15122-s25' ...
Creating directory /home/iliano/TMP/sample
Handout downloaded into assessment directory
Writeup downloaded into assessment directory

Due: Mon Jun 29 18:00:00 2020
sample successfully downloaded.
```

This creates a directory called `sample` with the following contents:

```
% ls sample
factorial.c0  favorite_number.c0  README.txt
```

The file `README.txt` tells you how to compile the assignment and how to submit it. The other files are what you'll be working on.

Editing your program

You can use any editor you wish to write and edit your programs, but we highly recommend you try out VSCode, vim, or emacs. These editors are very powerful and can do much more than just help you edit your code (as you will see).²

You can also use them from your own computer. To use vim or emacs, you will need to ssh in to one of the Linux andrew machines; once there, you simply run them from the terminal prompt as described below. To use the more modern VSCode, you will first need to install it on your computer by following the few easy steps in the *Guide to Success* on [VSCode on the course web site](#).

Nowadays, most people use VSCode, but you should practice using vim and/or emacs: if your laptop fails and you need to continue your work on a cluster computer, vim and emacs will be your only options.

- Using `cd`, go into the `sample` directory. From there open the file `factorial.c0` that you downloaded from the previous part of the lab:

VSCode:

Start the “Visual Studio Code” application and open the file `factorial.c0` from its “File” menu.

VIM:

```
% vim factorial.c0
```

Emacs:

```
% emacs factorial.c0
```

You should see the editor start in the Terminal or a VSCode window, and you should see a program that looks like it computes factorial³. The program is written in C0, the language we'll be using to start the semester.

²See the [course website](#) to learn more about using these editors.

³For **Emacs** or **vim**, trying to open a file that doesn't exist (i.e., by making a typo) will create a new empty file with that name. If you see a blank file, exit the editor and try again.

- ▶ Edit the program `factorial.c0` and add your name and section letter at the appropriate locations. Use the instructions below for the editor you're using.

VSCode: You can just start typing and use your mouse or arrow keys to move around the file. Insert your name and section letter in the appropriate comments in your program.

VIM: This editor has two modes – *insert mode* for inserting text or code, and *command mode* for entering 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, pressing "i" changes the editor to insert mode, allowing you to type text. Go into 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 won't need them right now. In the file, insert your name and your section letter in the appropriate comments in your program.

- ▶ Save your changes and exit the editor.

VSCode: To save your work, simply select "Save" from the "File" menu. If you use VSCode regularly, you will want to learn some of the shortcuts for common commands, for example Ctrl-s or Cmd-s to save a file.

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).

- ▶ 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 Linux command line.

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

- ▶ Be sure you're in the sample directory, then compile your `c0` 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.

- ▶ 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 `coin` 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.

- ▶ 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` and pressing Enter, or using the keyboard shortcut `Ctrl-d`.

Submitting Programming Assignments

- You will submit the assignment by typing the following at the Unix prompt:

```
% autolab122 handin sample factorial.c0 favorite_number.c0
```

The command to type is always written at the bottom of the file `README.txt` that comes with each assignment.

This command will ask you to acknowledge the academic integrity policy of the class and then submit the files to [Autolab](#). Once [Autolab](#) is done grading this submission, it will display your scores. Here's the full output:

```
% autolab122 handin sample factorial.c0 favorite_number.c0
Type "yes" to affirm that you have complied with this course's
academic integrity policy as defined in the syllabus: yes
Submitting to 15122-s25:sample ... (force)
Successfully submitted to Autolab (version 1)

Waiting for scores to be ready ...
Scores for 15122-s25:sample
  version | handin (1.0) | editing (1.0) |
+-----+-----+-----+
|      1 |          1.0 |          0.0 |
```

At this point, you can see the overall [Autolab](#) feedback with

```
% autolab122 feedback
```

Practice Problems

- You will submit practice problems using [Gradescope](#).

Practice problems are posted on [the handout tab of the course web page](#). In a browser, go to <https://cs.cmu.edu/~122q/handouts.shtml>, and click on “(UNGRADED sample)” under “Practice Problems”. Save a copy as `pp-sample.pdf`. You will write just your name and section number and then submit.

You can do the writing in two ways:

1. By entering annotations in `pp-sample.pdf` and then saving. The easiest way is to do so online by using [Kami](#) 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.
2. By printing `pp-sample.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 `pp-sample.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.

In-class Activities

Throughout the semester, we will do short online activities during lectures. Let's make sure you are all set.

► Point your browser to the [activity page](http://cs.cmu.edu/~122q/activities.shtml) (<http://cs.cmu.edu/~122q/activities.shtml>) and click on the **Test** button. Several things could happen:

1. You get an error message. Go back to the [activity page](http://cs.cmu.edu/~122q/activities.shtml) and follow the instructions under "Common misconfigurations". Then try again.
2. 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).
3. You get to a Google sign-in page. Authenticate with your **andrew email address**. Then proceed to step (d).
4. You are redirected to CMU's [web login page](#). Authenticate and proceed to step (e).
5. You see a Google form entitled "Testing your configuration" — good! Click on the radio button and then **submit**. You are ready to do activities!

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

At this point you are all set for 15-122!