15-122: Principles of Imperative Computation — Spring 2018

Lab 1: Setup — 9 January

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.

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.

(1.a) Open up a new Putty window to access the Linux command prompt.

On the Windows machines in the computer labs, 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:

- In the Hostname bar, enter "`YOUR_ANDREW_ID@unix.qatar.cmu.edu`"
- Keep the port number as **22**.
- Save these settings by entering a name (say, '*CMU*') in the box right under "`Saved Sessions.`" Afterward, hit "`Save`" on the right hand side.
- Next, click on the "`Open`" button.
- You will now see a new Terminal window open, and will ask you for your password.
- 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".

(1.b) 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.

(1.c) You should see the directory `private` as one of the entries in your home directory. Move to this directory using the cd command (cd 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.

(1.d) 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
```

(1.e) Download the factorial.c0 for this lab by running the following command:

```
% wget https://web2.qatar.cmu.edu/~srazak/courses/15122-s18/lab/factorial.c0
```

This will copy the factorial.c0 to your local directory.

(1.f) Verify you are in the `lab01` 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/lab01/
% 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`).

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

```
% /afs/qatar/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).

(2.b) Test that this works by **closing and reopening your terminal**, and typing

```
% coin -d
```

If the command above correctly starts the interpreter, then you're set up! **Don't forget to log out again before you leave.**

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.

(3.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.

(3.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.

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

EMACS: Once you're ready to exit, press Ctrl-x (the Control key and the "x" key at the same time) followed by Ctrl-c. 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'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`.

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.

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

(4.a) You will receive instruction via Piazza on how to submit Written Assignments. Carry on to the next section!

Programming Assignments

(4.b) You will submit your *programming* assignments (and some lab solutions) using Autolab. 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
```

If you are on a Windows machine, you would need another program to be able to view/download the files you are working on. 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 -> lab01) and download (click and drag) the tar file you just created.

Next, point your browser to Autolab(https://autolab.andrew.cmu.edu/courses/15122q-s18/), 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!**

At this point you are done with the lab, and you are free to go. **(4.c)** and **Part 5** are strictly optional (we really recommend you to do Part 5!).

(4.c) If you have your laptop, you can try to work through the first step of C0 at CMU, which can be found at `http://c0.typesafety.net/tutorial/C0-at-CMU.html`, *Connecting through SSH*, on your laptop.

Compiling and running your program

Before you can run your program, you need to compile it to have it translated into a lower level (machine) version of the code that can be executed by your computer. The *compiler* first checks for syntax errors and will abort with an error message in case it finds a syntax error. `cc0` or interpreter `coin` from the command line.

(5.a) Compile your code using the `cc0` compiler:

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

This runs the compiler with debug mode on (–d).

Debug mode checks all the code annotations starting with `//@` for testing. You will learn how these work 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 in your directory 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.

That shows us how to *compile* **(5.a)** and *run* **(5.b)** our programs. Alternatively, you can use the C0 interpreter to execute our program. An *interpreter* checks a program for syntax errors, translates it into machine code, and runs the resulting code step by step. This is a good way to interact with your program in real time to test it. The name of the C0 interpreter is `coin`.

(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 there a week from tomorrow. Be patient for now ☺.

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

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

In this case, you should see an annotation failure. This is because in our code, 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:

```
--> #quit
```

(5.f) Start the interpreter again, this time without the −d flag:

```
% coin factorial.c0
C0 interpreter (coin)
Type ‘#help’ for help or ‘#quit’ to exit.
--> factorial(-1);
```