Lab 1: Preliminaries & C Programming warmup

Objectives

- Review how to compile and execute C programs.
- Review basic C syntax.

You may complete this lab individually or with one colleague from this class using pair programming rules (see the course canvas page).

Preliminaries

(You may have already completed this step). In this class, we will learn to use a version of a Linux operating system and interact with it using shell. It will also come with a compiler that will let you compile C programs. There are several ways to obtain the system to work with. Mac OS X is built on a code very similar to Linux. So, if you have a Mac laptop, you can just go to Applications – > Utilities – > Terminal. Your shell is open, and you can skip to the next section.

An alternative is to get access to a remote Linux-like system. For now, you can create a free account on repl.it. Once you log in, create a blank workspace. Note that in the free account the workspace will be public. Create a new workspace (called Repl). This should allow you to run your code for this lab.

Or you can set up a virtual machine on your computer. Go to virtualbox.org and download for your platform. Go through the process of installing VirtualBox. [Mac users: If you get an installation failed message on installing VirtualBox, see Fixing 'The Installation Failed' VirtualBox Error on Mac High Sierra.] Download the Ubuntu virtual machine image (3GB) `os.ova` from here. Open it and it will install a new Ubuntu virtual machine for you in VirtualBox. Boot up the virtual machine and login. The default username is **student** with password **os**. To open a shell window, click on the "Terminal" on the left dock of the screen. A window will open with a purple screen with a prompt and a cursor. The prompt should look like: [*yourUserName@computerName*]$ Click the mouse point on the shell window to make it active. You are able to issue command to the computer using text. For example, type the following command followed by the Enter key.

```
date
```

You should see what you typed after the prompt and you should see the result of the command after you hit the Enter key.

Lab 1: Preliminaries & C Programming warmup

Note: the text of this lab assumes that you did the latter, local installation. You may have to adjust your steps if you chose one of the other two methods: basically, you won't be editing the files with gedit, you'll need to open the files with another text editor.

1 C programming

1.1 Hello, world!

Let's begin by writing, compiling, and running the canonical first program, Hello World!

```
#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  return 0;
}
```

1. Boot up your virtual machine and login.
2. Open up **gedit** (or another text editor). It is the "Text Editor" on the dock on the left-hand side of the screen.
3. Type the above source code for the C program and click "Save". For the name, type `hello.c`. For now, double-click on the folder "Documents, and click "Save.
4. Now open the **Terminal**. Just like **gedit**, it is located on the dock.
5. In the **Terminal**, you can type `pwd` and hit Enter to see your current directors (present working directory). Type `ls` to see a list of files and subfolders in your current directory.
6. You should see the Documents folder after typing `ls`. Type `cd Documents` to move ("change directory") to that folder. If you enter `ls` again, you should see your `hello.c` file.
7. Type `gcc hello.c` to compile the C program. Now when you enter `ls`, you should see a new file, `a.out`. Type `./a.out` to execute the program. You should see `Hello, World!` (If you want the name of the executable to be something other than `a.out`, you can use the `-o` option flag. `gcc hello.c -o hello` will create an executable with the name `hello`.

1.2 Code Samples

Lab 1: Preliminaries & C Programming warmup

The explanations for this and the following part are going to be less detailed than the preceding ones. Use your experience in learning your first computer language to approach learning coding in C. Remember to clarify all questions that emerge. Study the example source code files available on Canvas. The code contains comments detailing what each line of code accomplishes. Use this to get familiar with the basic syntax of C. Begin with `sample-hello.c`, then `sample-flowcontrol.c`, then `sample-function.c`.

1.3 Performance Statistics, lab1pr1.c

Design a program that lets the user analyze performance of a computing system. The user will enter the total time the system has been idle on each day of the week, in minutes. The program will store this information in an array. The program will then display the total idle time, the performance rate of the system during this week (i.e. percentage of idle time with respect to total length of the week), the average daily idle time, and the days with the lowest and highest load.

1.4 Hex verifier, lab1pr2.c

Write a boolean function `isHex(st)` that determines whether a given sequence of characters represents a valid hexadecimal number. We'll consider a sequence of characters to be a valid hex number if it is not empty and only contains digits and characters "A" through "F".

Write a program that asks the user to enter a character sequence and uses your function to perform the check. If the user enters a valid sequence, the program should just say "Thank you!". However, if the user enters something that's not a valid hex number, the program should keep asking the user until they enter a valid number.

For simplicity, you can assume that the number will be at most 20 characters long.

Submission

Upload your `lab1pr1.c` and `lab1pr2.c` to Canvas before the deadline.