Introduction

Computers are everywhere in modern day life: traffic lights use computers to adjust green time to roads that need it more; elevators use computers to dispatch cars efficiently; college students use computers to complete their assignments and surf Facebook; and many other examples! CPSC 1010 is one of the School of Computing introductory courses that teaches problem-solving and programming. That is, you will learn the fundamentals of creating software that runs on computers. CPSC 1011 is the required lab component of the CPSC 1010 course. Its goal is to give students additional practice in applying concepts learned in class.

Quite simply: the best way to learn programming is to "do" programming, which will involve learning and applying skills, and figuring out the best way to solve problems.

The purpose of today's lab is to introduce you to one way of writing a simple program using the C programming language while working in the Linux operating system.

Lab Objectives

By successfully completing today's lab, you will be able to:

- Log onto a School of Computing Linux computer.
- Use basic commands to navigate around a Linux command line prompt.
- Apply the general steps to create, compile, and run a simple C program.
- Submit your project files to a submission system.

Prior to Lab

- Please make sure you are properly registered for CPSC 1010 and 1011.
- We expect you to be comfortable and proficient using a computer: knowing how to use a mouse, enter in passwords, and use a Web browser such as Firefox or Chrome.
- We do not expect any prior programming experience in this class except for the basic understanding from lecture of the process of writing a software program and UNIX commands.
  - You should know what it means to compile and run a program from lecture.

Deadline and Collaboration Policy

- This assignment is due by 11:59PM on Friday (8/30/2019).
- Use Canvas to submit your code for Exercises 1, 2 & 3.
  - More instructions on what and how to submit are included at the end of this document.
- You should write your solutions to the exercises in this lab by yourself. It is okay to talk, at a high level, with someone else in the class about a solution. However, you should not show or discuss your code to anyone but a course instructor or lab teaching assistant.

Lab Instructions

Part 0: Understanding Formatting of CPSC 1011 Labs

We will strive to be consistent as much as possible in the structure of labs, especially in how we use formatting to provide additional context. For instance,

- Key terms will be typically **bolded** in style. These are usually important terms that tie back to concepts that were talked about in lecture.
- The names of files (e.g., **filenames)** will be shown in a mono-spaced font, such as Andale Mono.
- Instructions that you will give a computer (e.g., **commands**) and **code** (e.g., a series of commands you have programmed your computer to execute) that you write will be shown using a mono-space font, such as Andale Mono and have a light orange shading behind:

  Code Example:

```
int main() {
        printf("This is how code will look!");
        return 1;
}
```

  Command Example:

```
prompt% echo "This is how a command will look!"
This is how a command will look!
```

  As a quick note, the "prompt%" in the box above is used to indicate the generic **prompt** a user sees in the terminal. This is often different depending on the type of operating system and terminal you have configured, and can be changed depending on your preference. On a School of Computing machine, it will likely look like what is below. You should NEVER type in the "prompt%" no matter what it looks like!

```
[17:21:02] yourClemsonUsername@access1:~ [1] scp examplefolder/examplefile.txt .
```

- Submission instructions will always be placed at the end of the lab document, to coincide with the thought process that most students undertake. We may be using multiple submission systems this semester, so please make sure that you do not wait until the last minute to hand in your lab!
- In most cases, specific exercises that you will be completing as part of the lab will be show with a purple background:

  Demo Exercise

  1. How much wood would a woodchuck chuck if a woodchuck could chuck wood?

Now, onto the real work!

Part 1: Using a Linux Workstation

Many of the School of Computing's computers use an operating system called **Ubuntu**, which is a version of **Linux**. Ubuntu is a free and open-source operating system with millions of users worldwide. What does that really mean? Ubuntu is free in that there are typically no costs to buy the operating system and people from all over the world help refine, improve, and enhance Ubuntu. No single company has control over the Ubuntu or

Linux. Furthermore, open source communities rely on skills of individuals in addition to programming, such as graphical designers, artists, musicians, and technical writers. Linux is the name for a family of operating systems that operate like a **UNIX** operating system, a powerful and reliable operating system that Bell Laboratories created in 1969.

Confused? Don't worry about it—the family tree for Linux/UNIX is pretty complicated. Oh, Macintosh OS X is another flavor of a UNIX operating system. If you have an Android phone, there's a Linux operating system there, too!

For the intents of this course, we will be looking at aspects that are pretty much common across these different variations. Some will have different **graphical user interfaces** (much like how Macintosh OS X looks different than Windows 10, but a pro user on one of those platforms would still be able to figure out and accomplish tasks using another system.)

So why use Linux or UNIX? These systems tend to be very popular in "production environments" where organizations are creating products due to the reliability and stability of the interface. If you intend on getting a job as a software engineer, there's a pretty a good chance you will be working in Linux or UNIX environment.

By now, you've likely logged onto your computer workstation using your Clemson username and password. If you haven't, please type your Clemson username into the login window and then type in your password when the computer prompts you to.

- For the time being, ignore the "Language", "Session", "Restart", and "Shutdown" options.

Once you've logged onto your workstation, you will see a computer desktop, which looks similar to a desktop that you would see on Mac OS X or Windows 10, but with different icons and slightly different terminology. Spend some time exploring the interface.

<u>Lab Exercise 1</u>

Type your answers to each of the questions below in the lab01.txt document that you create for Step 2 of this exercise.

1. How many different text editors (e.g., programs that you can type text into and save files) can you find installed on your workstation? List them.
2. Using one of the text editors you found in Step 1, create a document called lab01.txt and save it to the desktop. In the top left of the document, please put the following information:
   a. Your first and last name
   b. Your CID
   c. Your lecture and section and your lab and section
   d. Lab 01 – Exercise 1

   For example:
   Thomas Clemson
   C1234567
   CPSC 1010-003 and 1011-001
   Lab 01 – Exercise 1

3. How would you delete a file?

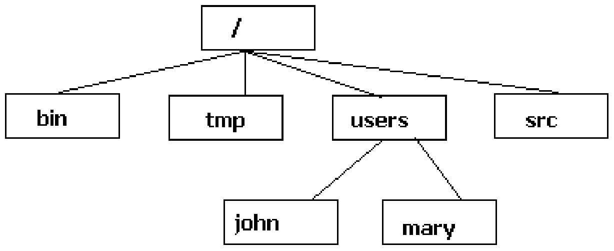
4. Which web browsers are available? Which ones support Java and/or Flash? (How would you determine this?)

5. Which applications could you use to edit an image?

Later, you will submit your lab01.txt document to Canvas as part of your lab submission.

Part 2: Filesystems

One of the most popular ways that information (e.g., **files**) is organized on computers is by using a **hierarchical filesystem**. Generally, files are stored on a computer using a tree format having a **root** and **leaves**, where the root is the starting part of a tree and the leaves are end nodes.

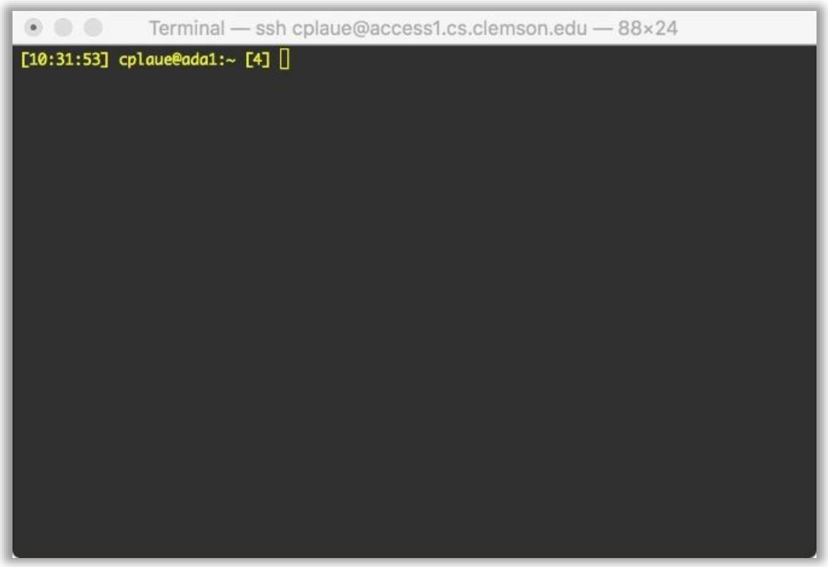


As you maneuver through a tree, you are **traversing** through it. The current place that you're at while traversing the filesystem tree is called your **current working directory**. Linux/UNIX operating systems use a hierarchical approach to files, as you will explore.

Part 3: Navigating in a Command-Line Interface

One powerful feature of Linux and UNIX operating systems is the ability to do most things via keyboard commands. We call this type of way of interacting with a computer as using a **command-line interface**. In your CPSC 1010 lecture, your instructor has likely demonstrated several of the commands.

In this part of the lab, find the **Terminal** application and open it. You will have a window that looks like this:



```
[10:31:53] cplaue@ada1:~ [4]
```

You can enter in **commands** via the keyboard and press enter to run them. At a basic level, commands are only verbs---you are telling the operating system to do something. Some commands take a verb and an object, some commands take a verb and two objects. For instance, if you used the command cp to copy a file, you would need to also list the name of the file you want to copy and the name that you are copying to:

```
cp myfile myfile2
```

The following table lists several frequently used commands in Linux/UNIX:

Command	Description
pwd	Prints the **working directory** (e.g., shows you where you are currently in the filesystem)
mkdir	Makes/creates a new directory
cd	Changes/moves into a directory
ls	List the contents of the current directory
rm	Delete a file
rmdir	Delete a directory (but the directory must be empty)

Before we have you exploring the command-line interface, you need to know a few more things to put together.

Item	Description
.	A single period (usually called a dot) represents the current directory
..	Two periods (usually called dot dot) represents going back one directory
/	Forward slashes are used in-between directories
~	Tilde is used usually to represent your **home directory**, which is the main starting point (e.g., root) for your account on the Linux workstation.

Lab Exercise 2

Add your answer to this question as #6 in the file lab01.txt that you created in Part 1 of this lab

- One practice of a successful programmer is to be organized. What better place to start than to making sure our files are well-organized? What are the exact sequence of commands that you need to create the following filesystem tree?:

  home -> cpsc 1011 -> lab01 -> inlab
  (Consider every key that you need to press!)

- Make sure you create this directory structure for the next part of this lab assignment

Part 4: Creating Your First C Program.

In your inlab directory that you created in the last part of this lab, you will create and run your first C program! **C** is a language developed in Bell Labs and has been around for a long time, but it's incredibly powerful and a great starting point for learning programming fundamentals. But first before we get started writing lab00.c, we need to have a text editor that you can write your program in. You can simply launch the program with a command. Try out these commands in order:

```
gedit

gedit &
```

Note when you run the gedit command without the use of an ampersand (&), you aren't able to run any more commands at the prompt. The ampersand tells Linux to start gedit as a program, but also allow you to continue typing commands. You'll learn more about what's going on underneath the hood in future classes.

In your gedit window, *type* (do not copy and paste) the following text (which is your first program):

```
#include <stdio.h>

int main()
{
        printf("Hello, world!!!!!\n");
        return(0);
}
```

Save the file as lab00.c and return to the terminal.

To **compile** your first program, enter the following command:

```
gcc lab00.c
```

If everything compiles properly, you will see the prompt in the terminal. You can use the ls command to verify that the file a.out is now in the current directory.

If there's a mistake in the program, you will see errors or warnings be displayed on the screen. If you see errors, make sure that you've typed the program exactly as displayed above.

To run your program, you will use the following command:

```
./a.out
```

Recall, the single . means that you are in the current directory.

Thus, at this stage, you should see Hello world!!!!! on the screen in front of you.

Lab Exercise 3

- Create a new file called lab01.c
- In this file, modify your simple Hello World Program to print out exactly the following:

```
Hello, world!!!!!
Where the Blue Ridge yawns its greatness,
Where the Tigers play,
Here the sons and daughters of dear old Clemson,
Program labs all day!
```

Submission Guidelines

- Submit your lab01.txt answers to the Canvas assignment. Make sure that you have the heading as specified and that you have your answers numbered.
- Submit your lab01.c program to the Canvas assignment.

Grading Rubric

- If you are not present and attending lab on Tuesday and Thursday, you will not receive credit for this lab.
- If both parts of the lab assignment are not submitted on time to Canvas, you will receive a 0 for the assignment.
- Tentative point distribution

Correctness and sufficiency of answers in lab01.txt	40 points
Following documentation best practices (code & writeup)	10 points
Correctly functioning lab01.c program	50 points