CS 2400
Laboratory Assignment #1: Exercises in Compilation and the Linux Programming Environment
(100 pts.)

1 Introduction

This laboratory is intended to give you some brief experience using the editing/compiling/file management tools that are available in the Linux environment. You will not be required to do any actual programming in this laboratory exercise. However, you will need to follow directions exactly in order to complete the laboratory successfully. While there are usually two ways to accomplish the tasks laid out in this laboratory (either via the windowing environment via point and click or from the command line), we will use the command-line approach for most of this lab.

2 Getting Started

Step 1: Obtain your username and password from laboratory teaching assistant. Do not confuse this username/password with your OHIOID and OHIO password, the ones that you use for your email account and for Blackboard. You will need both to successfully complete this laboratory.

Step 2: Log on to the workstation that you are sitting at using your CS username/password. When you login for the first time, you may see an error message relating to the desktop. Do not worry about it. Simply click OK.

Step 3: Once you have logged in you will see a toolbar on the left with some icons which look familiar to you, including one for the Firefox browser and one for File management. The circle at the top is the Search. Click on this. In the search text box type **term** to pull up the Terminal icon. Then, using the mouse, drag and drop this icon onto the toolbar at the left. Now the Terminal will be readily available every time you log into one of the lab machines. (You do not need to log into the same machine every time because your desktop configuration and your files are stored on the server and will "follow" you from machine to machine.)

Step 4: Click on the Terminal icon on the toolbar and continue with this lab.

3 Linux Fundamentals

Linux is an operating system just like Microsoft Windows and Mac OS. Linux is a multi-user operating system. Each machine that you are sitting at may have multiple users. For these laboratory machines, normally, only you will be logged into them. In order to see which other users are using the machine that you are logged into, type

```
who
```

at the terminal prompt and hit "enter." The Linux command shell (the terminal window that you are using) and the windowing environment allow you (i) execute programs, and (ii) view and manipulate files. The Linux file system consists of directories and regular files. Directories

are simply a "folder" where you place a collection of files. Directories may contain other directories. These other directories are subdirectories of the bigger folder (parent). The root of the entire file system is the directory `/`. The root of your directory is usually `/home/username`, where username is your username.

3.1 Listing the Directory

From the terminal, you can display all of files (and subdirectories) in your current directory by typing

```
ls
```

(You can also look at your current directory by using the windowing system. It's useful to know how to do both.) With the command `ls` you can list all of the files matching some pattern in your directory. The wildcard character `*` can be used to denote any string of characters. So, issuing the command `ls *.cc` will list all of the files in the current directory with the `.cc` extension. Similarly, you can give `ls` the name of a directory to list. Issue the command `ls /` to see what happens.

3.2 Working Directories

The Linux shell always keeps track of where you are in the file system. In particular, it "remembers" the location of the current directory that you are working in. This is your "working directory." To see your working directory, type

```
pwd
```

This is the command for "print working directory."
There are two "special names" for directories in the Linux shell. The first special name is "`.`", which denotes the current working directory. The second name is "`..`" which denotes the parent of the current working directory.

3.3 Changing Directories

You can change between directories using the command `cd` in the Linux shell. To change to a specific directory, you can issue the command "cd directory". For example, `cd /` changes the directory to the root of the file system. To get back to your home directory, you can always type `cd` by itself. To get to the parent directory, simply type `cd ..` The command `cd .` does nothing (why?).

3.4 Creating Subdirectories

You can create subdirectories by issuing the command `mkdir`. For example, the command `mkdir CS2400` will create a subdirectory in your home directory called `CS2400`.

3.5 Copying, moving, and deleting files

The commands `cp`, `mv` and `rm` copy, move, and delete files in your directory.

3.6 Manual Pages

To find out more information on a given Linux shell command, simply type

```
man command
```

For example,

```
man cp
```

displays the manual pages for the `cp` (copy) command.

3.7 Additional Linux Commands

Some other basic Linux commands that you can enter from your terminal:
• `rm` — delete a file.
• `mv` — move (rename) a file.
• `more` — display a file.

3.8 Lab Answer Sheet

After reviewing the previous sections, answer questions 1–8 on the lab answer sheet. Don't forget to write your GitHub user name.

4 Change Your Password

Place your cursor on the "terminal window". One of the first commands that might want to use is the following: Change your password by using the `passwd` command.

```
% passwd
passwd: Changing password for example
Enter login(NIS+) password: (ENTER YOUR OLD PASSWORD)
New password: (ENTER YOUR NEW PASSWORD)
Re-enter new password: (RE-ENTER YOUR NEW PASSWORD)
NIS+ password information changed for example
NIS+ credential information changed for example
```

Contact John Tysko if you ever forget your password. tysko@ohio.edu

5 Create Directories

The prompt that is being displayed will be the name of the workstation at which you are sitting. To keep your space tidy we will begin by creating some directories.

Type **`mkdir 2400`** *make the directory 2400 under the current working directory*
Type **`cd 2400`** *go into the new directory*
Type **`mkdir Labs`**
Type **`cd Labs`**
Type **`mkdir lab1`**
Type **`cd lab1`**

This directory is where you should perform all of your work for this lab.

6 Useful Linux Constructs

The Linux shell has a number of useful features that allow you to manipulate the input and output of files.

I/O Redirection:

Redirecting the standard output. You can redirect the output of any Linux command to an output file (`file.out` in this example) instead of to the screen by placing the symbol "`>`" in front of the output file name. For example, the following command will output a list of the files in your current directory and save them to the specified file:

```
ls > file.out
```

Redirecting the standard input. You can redirect your program to take its input from a file (`file.in` in this example) instead of from the keyboard by using the input redirection symbol "`<`".

```
     myprog < file.in
```

Pipes:

```
ls | more
```

Script:

```
> script record.out
Script started, file is record.out
> ls
notes1.aux notes1.tex~ notes2.tex
> exit
exit
Script done, file is record.out
```

```
> more record.out
```

7 The C++ Compiler

The C++ compiler in our Linux environment is free software developed by the Free Software Foundation (GNU).

The program is `g++`. To compile a C++ program using `g++`, simply type

```
g++ -Wall program.cc
```

If the program file `program.cc` is complete and correct, the `g++` compiler will produce an executable file called `a.out`. To run your program, simply issue the command

`./a.out` (Note that `.` is the current directory.)

The `g++` compiler has many options. To see them, issue the command `man g++`. You may ignore most of these options for now. However, the following options to `g++` may be useful to you.

- `-g`: This option compiles your program with useful debugging information included.
- `-o` outfile: This option places the result of the g++ in the file outfile. This is one way to change the name of your executable from a.out to something else.
- `-c`: This option stops compilation at the object file level. We will use this option later.
- `-O`: Optimize the code. This option is incompatible with the -g option.
- `-Wall`: This option shows the user all warnings messages that the compiler detected.

7.1 Lab Answer Sheet

Complete questions 9 – 11 on the lab answer sheet.

8 Problem Solving

The next few problems will give you some practice using the Linux editors, the compiler, and some other useful commands.

Writing and compiling a simple C++ program

First, go to the `lab1` directory you created. Use an editor that you feel comfortable using. Good choices are: `code, atom, nano, gedit.` In the terminal window you can type:

```
code firstProgram.cc
```

This will start the VSCode editor with the file name specified.

All real programmers meet a new programming environment with a "Hello" program. We will now write such a program in our new environment.

```
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//
//   program:  <File Name>
//
//   Name   :   <Enter your name>
//   Date   :   <Enter today's date>
//   Email  :   <Enter your e-mail address>
//
//   Description: This program prints a message
//                to the screen.
//
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

#include <iostream>
using namespace std;

int main( )
{
    cout << "Hello Me! \n How am I today? \n";
    //You can be creative with your message.
    return (0);
}
```

1. Name the file whatever you wish—an appropriate choice would be **`lab1.cc.`** You can use any one of the following extensions: .cc, .cpp, or, .cxx

2. Compile `lab1.cc` by using the command `g++ -Wall lab1.cc`

3. Issue the command `ls` and **answer question # 12 on the lab answer sheet.**

4. Run the executable by typing `./a.out` (see the output on the screen)

5. Use I/O redirection to save the results of this command as follows. You can redirect the program output to a file called `lab1.out` instead of to the screen.

   `./a.out > lab1.out` **(warning: do not redirect to lab1.cc)**

6. Issue the command `ls` and **answer question # 13 on the lab answer sheet.**

7. Type `more lab1.out` to make sure that you produced the correct output.

8. Create a GitHub account using your school's email and edit your profile to enter your full name.

9. **Answer the remaining questions on the lab answer sheet.**