CSCE 155 - C

Lab 1.0 - Introduction

Cole Peterson

Prior to Lab

In each lab there may be pre-lab activities that you are *required* to complete prior to attending lab. Failure to do so may mean that you will not be given full credit for the lab. For the first lab, there are no pre-lab activities.

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- The general lab environment & CSE system including its policies and expectations
- Basic unix commands
- Retrieving lab code from Github
- Modifying, compiling and executing your first C program
- Using CSE's web handin and web grader

2 Activities

2.1 Login & Consent Form

You will receive your login from the CSE System Administrators who will also give you an overview of the CSE system and its policies. Be sure to login and change your temporary password. Some departmental resources that you may find useful:

- CSE Website: http://cse.unl.edu
- UNL Computing Policy: http://www.unl.edu/ucomm/compuse/
- CSE Academic Integrity Policy: http://cse.unl.edu/academic-integrity-policy
- CSE System FAQ: http://cse.unl.edu/faq
- Account Management: https://cse-apps.unl.edu/amu/
- CSE Undergraduate Advising Page: http://cse.unl.edu/advising
- CSE Student Resource Center: http://cse.unl.edu/src

2.2 Lab Introduction

- Lab instructors and TAs
- Office Hours
- Student Resource Center (Avery 12)
- Lab policies

2.3 Basic Unix Commands

Much of your work will be done on the command line in the Terminal application. Take a moment to familiarize yourself with the following basic commands. A more comprehensive tutorial on unix commands is available here: http://www.math.utah.edu/lab/unix/unix-commands.html.

- Show the current working directory: `pwd`
- Creating a new directory: `mkdir dirName`
- Changing directories: `cd dir_name`
- Moving up a directory: `cd ..`
- Moving directly to your home directory: `cd ~`
- Listing files in a directory: `ls`
- Listing details of files in a directory: `ls -l`
- Listing files in another directory: `ls dir_name`
- Listing the contents of a (text) file: `more file_name`
- Removing files (careful!): `rm file_name`

Text Editors

Programming requires that you write code in a source file: a plain text file that contains syntactically valid programming commands. In general, any plain text editor can facilitate this (MS Word is not a plain text editor). Atom is already installed on your VM already. The lab instructor will demonstrate some of the following options:

- jpico
- emacs
- Notepad++ (a graphical editor launched from Windows)
- Atom.io https://atom.io/
- Sublime Text https://www.sublimetext.com/

2.4 Checking Out Code From Github

Each lab will have some starter code and other *artifacts* (data files, scripts, etc.) that will be provided for to you. However, the code is hosted on Github (https://github.com) and you must check it out. You will not need to know the details of using git nor be a

registered Github user to get access to the code necessary for your labs or assignments. However, you are *highly encouraged* to learn this essential tool. You may find it very useful to keep track of your own code and to share code if you work in pairs or groups.

To check out the code for this lab, do the following.

1. If you haven't already, install VMWare Horizon Client and connect to the VM pool with your CSE login
2. Open the Terminal application
3. In your home directory, create a new directory for all your labs:
   `mkdir labs`
4. Verify that this worked by listing the contents of your directory by typing `ls`. You should see your `labs` directory.
5. Go to this directory by changing your directory:
   `cd labs`
6. "Clone" this lab's code by using the following command:
   `git clone https://github.com/clptrsn/CSCE155-C-Lab01`
7. A new directory, `CSCE155-C-Lab01` should be created, go to this directory by typing `cd CSCE155-C-Lab01`.
8. List the contents of this directory by typing `ls`. If everything worked, there should be a `hello.c` file in your directory.

2.5 Your First Program: Hello World!

As this is an introductory course, most of the C programs you will write in this course will execute and require interaction from the command line rather than a more user-friendly graphical interface.

The code you cloned contains a basic "Hello World" program that, when compiled and run will simply print the message "Hello World" and exit.

To compile and run this program, do the following.

1. Type `gcc hello.c`, this *compiles* the source code into an *executable* file named `a.out`. Use `ls` to verify that the new file has been created.
2. Run your program by typing the following: `./a.out`

Let's now modify the program.

1. Open the `hello.c` source file in the text editor of your choice (Notepad++ is recommended for beginners). Your program should look something like the following:

```c
/**
 * Author: Chris Bourke
 * Date: 2016/11/02
 *
 * A simple hello world program in C
 *
 */
#include<stdlib.h>
#include<stdio.h>

int main(int argc, char **argv) {

  printf("Hello World!\n");

  return 0;
}
```

2. Modify the file by changing the author to you and your partner and change the date.
3. Change the message that is produced by the program (the `printf` statement) to instead print you and your partner's names.
4. Save and exit your editor and repeat the process to compile and run your program and verify that your changes have taken effect.

3 Handing In & Grading

Many of your assignments will include a programming portion that will require you to hand in *soft-copy* source files for graders to compile and evaluate. To do this, you will use a web-based handin program. After handing your file(s) in, you can then grade them by using the web grader. To demonstrate, do the following.

1. Open a browser to https://cse-apps.unl.edu/handin
2. Login with your CSE credentials
3. Click on this course/lab 01 and handin the `hello.c` file. You can either click the large "handin" area and select the file or you can drag-drop the file. You will be able to re-handin the same file as many times as you want up until the due date.
4. Now that the file has been handed in, you can "grade" yourself by using the webgrader; open a new tab/window and point your browser to one of the following depending on which course you are in:

- https://cse.unl.edu/~cse155a/grade
- https://cse.unl.edu/~cse155e/grade
- https://cse.unl.edu/~cse155h/grade

5. Fill the form with your CSE login and password, select the appropriate assignment (Lab 01) and click "Grade"
6. For future assignments and labs, you can compare the results of your program with the "Expected Results". If there are problems or errors with your program(s), you should fix/debug them and repeat the handin/grading process. You can do this as many times as you like up until the due date. Some programs and assignments will run test cases and may provide expected output alongside your output. Others may have more sophisticated test cases and actually provide you a percentage of test cases passed. It is your responsibility to read, understand and *address* all of the errors and/or warnings that the grader produces.

Demonstrate your working programs to a lab instructor, answer all the questions on your lab worksheet, and have them sign off on it.