CSE31 : Lab #0 - Introduction to C

Overview

Welcome to CSE31! This lab will help you be familiar with the C programming language and the tools available to you on the Linux systems in the lab. In this exercise, you should do your work on the lab systems directly so to minimize any headaches in the beginning of this semester. After you are comfortable with the system and the language would be a better time for exploring your other options. The goal of this lab is to create simple programs that will display output to the console, read input from the user, use if statements and for loops in C, and perform some simple error handling.
Note: You need to have a separate program for each of the parts of this lab. When you submit your assignment through CatCourses, make sure that ALL PARTS are included.

Getting started

In this class, we will use the Linux terminal extensively, so you should get familiar with it as much as possible (there is plenty of information online). We recommend setting up a smart directory structure to use throughout the class, but it is up to you how you setup your programs. We would setup a CSE31 directory on the Desktop, a directory for the lab (i.e. Lab_0) inside it, and a directory inside the lab directory for each part of the lab (i.e. part1, part2, etc...).

(Exercise) Tutorial in Linux

If you are not familiar with the Linux operating system or not sure how to use command lines in a Linux terminal, take a look at the tutorial from the following link during your lab hours. Otherwise, skip to the next exercise.
http://linuxcommand.org/lc3_learning_the_shell.php

(Exercise) Create – Lab_0 directory

Open a terminal and access your Desktop directory. To create a new directory, use ***mkdir*** command. Make sure you are in your Desktop directory, type ***mkdir CSE31*** and press enter in the terminal. To verify if the directory is created, type ***ls*** (lowercase L, not uppercase i) and press enter. You should see **CSE31** among a list of directories and files under your Desktop directory. Go into the newly created CSE31 directory by typing ***cd CSE31*** and press enter. Now, create a new directory named **Lab_0** under your current (CSE31) directory. Go into the newly created Lab_0 directory and start working on your lab assignment.

(Exercise) Create – main.c

Make sure you are in your Lab_0 directory, type ***gedit main.c*** and press enter. The **gedit** text editor will be displayed. Feel free to use any other text editors (vi, nano, etc.) of your choice. Ask your TA if you are not sure about this.

Copy the following code to your file:

```
#include <stdio.h>
int main()
{
   printf("Welcome to CSE 31!\n");
   return 0;
}
```

Save your file and exit gedit.

Congratulations, you've just written your first C++ program!
The first line, **#include< stdio.h>**, includes a library that allows you to use simple Input/Output (I/O) operations. In this program, it allows you to use the **printf** statement. In the next line, **int main()**, we start the main function, which is a mandatory function for any C program. This is the function that first executes when the program is launched. The contents of the function's code need to be encompassed by curly braces (lines 4 and 7). On line 5, we are printing the text **Welcome to CSE031!** to the screen, ending with a new line. Finally, the last line returns from the main function, which will stop execution of the main program.

Once you have created this file and understood its content, you want to compile the source file (main.c) into an executable so that you can run it on your computer. You will do this from the command line by running:

```
g++ <source> -o <executable>
```

where **gcc** is a program (already installed on your Linux system) that compiles C source files, **<source>** is the source file for your program (main.c in this example), **-o** tells the compiler that you want to give the executable its own name, and **<executable>** is the name you want to give your program. In the terminal (still under Lab_0 directory), compile your first program by typing ***gcc main.c -o main***

Assuming that your program compiled successfully (i.e. no errors were found), you can run your program by typing **./main** in the terminal.

(Assessment) Logic Check

It is crucial that you gain a thorough understanding of creating and compiling programs. Answer the following questions (either try to answer them now or in your own time, but make sure you can answer them):

1. How would you change the executable name from **main** to **cselab**?
2. What happens when you compile using **gcc <source>** only?
3. Add comments to your program.
4. Play around with "printf" statements and make sure you understand how to print out values of different variables.

Write your answers to questions 1-3 in a text file for submission.

(Exercise) Create – hello1.c

In this part of the lab, you should create a program that outputs **Hello World!** to the screen. Please note that this program should be very similar to the main.c. When you run your program, it should display the following output:

```
Hello World!
```

(Exercise) Create – hello2.c

In this part of the lab, we will introduce user input. More specifically, we want to welcome a specific student to the CSE031 class. In order to do so, you should start by asking the user for his name, by outputting **Please enter your name:** to the screen. You will then read the student's name, which could be his first name, full name, both, or any other string that is typed before pressing enter (see examples in the next sections). After you have read and stored the student's name, you will output **Welcome to CSE031, <name>!** where **<name>** is the name that the user inputted. Note that there is NO string type in C. Refer to the online tutorial on how to use ***scanf*** for user inputs
Web address of the tutorial:
https://www.tutorialspoint.com/c_standard_library/c_function_scanf.htm

Here are the examples of the output (input is in italic and bold):

Please enter your name: ***Name***
Welcome to CSE031, Name!

Please enter your name: ***First Last***
Welcome to CSE031, First Last!

(Exercise) Create – punishment1.c

A common punishment for school children is to write out the same sentence multiple times. Write a C program (**punishment1.c**) that will write out the following sentence a number of times inputted by the user: **C programming language is the best!** The program will first ask the user to input the number of times the sentence should be written, by outputting the following sentence to the screen: **Enter the number of lines for the punishment:** . You should check to make sure that the value entered is correct (think about what would constitute an incorrect value). If the value entered is incorrect, output the following sentence to the screen: **You entered an incorrect value for the number of lines!** and stop the program. If a correct value was entered, display the sentence **C programming language is the best!** the number of times inputted by the user.

Here are the examples of the output (input is in italic and bold):

```
Enter the number of lines for the punishment: 5
C programming language is the best! C programming language is the best! C programming
language is the best! C programming language is the best! C programming language is the best!
```

```
Enter the number of lines for the punishment: -5
You entered an incorrect value for the number of lines!
```

(Exercise) Create – punishment2.c

Since we want to make the punishment more realistic, we will add a typo in one of the lines of the punishment. This is very similar to what you have already done in punishment1.c, so it would be a good idea to copy your punishment1.c and start from there. Your new program (punishment2.c) will ask the user for the number of lines for the punishment: **Enter the number of lines for the punishment:** . Once again, if an incorrect value has been entered, your program should output **You entered an incorrect value for the number of lines!** and stop executing (up to this point, you should not have to change anything from your last exercise. Next, the program will ask on what line a typo should be made by outputting **Enter the line for which we want to make a typo:** to the screen. Once again, you should check that the value entered by the user is correct (think about what would constitute an incorrect value). If the value is incorrect, display **You entered an incorrect value for the line typo!** and stop program execution. If both inputs are correct, you should then display the punishment sentence the correct number of times (**C programming language is the best!**), making sure to change it to **C programming language is the bet!** (the typo) for the line number defined by the user/input.

Here are the examples of the output (input is in italic and bold):

```
Enter the number of lines for the punishment: 4
Enter the line for which we want to make a typo: 1
C programming language is the bet! C programming language is the best! C programming
language is the best! C programming language is the best!

Enter the number of lines for the punishment: 6
Enter the line for which we want to make a typo: 3
C programming language is the best! C programming language is the best! C programming
language is the bet! C programming language is the best! C programming language is the best!
C programming language is the best!

Enter the number of lines for the punishment: -8
You entered an incorrect value for the number of lines!

Enter the number of lines for the punishment: 3
Enter the line for which we want to make a typo: -2
You entered an incorrect value for the line typo!
```

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have included the following ***before*** you press Submit:

- Your main.cpp, hello1.c, hello2.c, punishment1.c, punishment2.c, answers to Assessment questions (1-2) in a text file, and a list of Collaborators.