CSE31: Lab #1 - 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, it would be a good time for exploring your other options.

The goal of this lab is to refresh your programming skills and create simple programs that will display output to the console, read input from the user, use conditional and loop statements 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 save your assignments throughout the semester, but it is up to you about how you want to save your work. We will setup a CSE31 directory on the Desktop, a directory for the lab (i.e. Lab_1) inside it, and a directory inside the lab directory for each part of the lab (i.e. part1, part2, etc...).

You must show your answers to the lab activities before leaving lab in order to receive participation score.

Tutorial in Linux

As a software engineer (or normal human), it is impossible to remember all the commands for Linux terminal. It is particularly true for those that you don't use very often. So, where can you find the resources?

TPS (Think-Pair-Share) activity 1 Paired with the classmate sitting next to you and do the following tasks (you are allowed to have groups of 3 students):

1. **Record your partner's name.**
2. Independently search from the internet 3 online tutorials on how to use Linux terminal.
3. Share with your partner what you have found.
4. Bookmark your results in the browser of your computer.

Create Lab_1 directory

TPS activity 2 Paired with the same classmate sitting next to you and discuss questions 1 – 4 (10 minutes):

1. How can you open a terminal from your Linux computer?
    a. Can you open more than 1 terminal at the same time?
    b. Why do you think you want to open more than 1 terminal at the same time?
2. In the terminal, how can you tell what are inside the current directory?
3. From your current directory, how can you navigate to ***Desktop*** directory?
4. While you are in ***Desktop***, let's create a new directory called ***CSE31***. How to do it?
5. Your TA will "invite" one of you randomly to share what you have discussed.
6. Now navigate to the directory you've just created (***CSE31***), create another new directory called ***Lab_1*** here (***Lab_1*** is inside ***CSE31***)
7. You will be saving all the programs you create today in this directory.

(Exercise) Create – main.c

Make sure you are in your Lab_1 directory, type ***gedit main.c*** and press enter. The **gedit** text editor will be displayed. Note that **gedit** is only available in a Linux system with graphic interface. You may want to use other text editors (vi, nano, etc.) if terminal is the only interface to access a Linux system. Ask your TA if you are not sure about this.

Copy the following code to your file:

```
1    #include <stdio.h>
2    int main()
3    {
4        printf("Welcome to CSE 31!\n");
5        return 0;
6    }
```

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 3 and 6). On line 4, 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. To compile, we will use GCC compiler (not G++, it's for C++).

TPS activity 3 Paired with the same classmate(s) sitting next to you and discuss questions (15 minutes):

1. Independently find 2 online references on how to use GCC in a Linux terminal.
2. Share with your partner what you have found and save your results in the bookmark of your browser. You will refer to these references to answer the following questions.
3. What command do you type in the terminal to compile your **main.c**?
4. How do you know if your file is compiled successfully?
5. What does the **–c** flag do in gcc?
6. What does the **–g** flag do in gcc?
7. How do you change the executable name from main to cselab1?
8. What happens when you compile by typing "***gcc main.c***" only?
9. Now, let's run the program you've just compiled. What command do you use?
10. Your TA will "invite" one of you randomly to share what you have discussed.

(Assignment 1, individual) Create –punishment.c

A common punishment for school children is to write out the same sentence multiple times. Create a C program (**punishment.c**) that will write out the following sentence the number of times specified by the user: ***"C programming language is the best!"*** Your program will ask the user for the number of lines for the punishment: ***Enter the number of lines for the punishment:*** If an incorrect value has been entered, your program should output ***You entered an incorrect value for the number of lines!*** and stop executing. To make this program "realistic", it will introduce a typo in a certain line. It will ask in 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.

You will need to use ***scanf*** to process user inputs. Look it up online to find out how to use it.

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!

(Assignment 2, individual) Create –averages.c

Create a new program that will ask a user to enter a number repeatedly. This program will calculate 2 averages based on the sign of the inputted numbers: ave_pos (average of all positive numbers) and ave_neg (average of all negative numbers). The program will stop when the user enters a '0'.

Before writing the program in C code, write a pseudocode to describe your approach to this problem.

TPS activity 4 Paired with the same classmate sitting next to you and write the pseudocode together (15 minutes). **Pseudocode only. You must write the C code individually.** Your TA will "invite" one of you randomly to share your pseudocode.

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

Please enter an integer: ***1***
Please enter an integer: ***-1***
Please enter an integer: ***2***
Please enter an integer: ***-2***
Please enter an integer: ***0***
Positive average: 1
Negative average: -1

Please enter an integer: ***-1***
Please enter an integer: ***-2***
Please enter an integer: ***-3***
Please enter an integer: ***0***
Negative average: -2

What to submit

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

- Your punishment.c, pseudocode for averages.c, averages.c , answers to the TPS activities in a text file, and a list of Collaborators.
- Your assignment is closed **7 days after this lab is posted** (at 11:59pm).
- You must demo your submission to your TA **within 14 days** (preferably during next lab so you will have a chance to make correction and re-submit/re-demo.)