ITF22519: Introduction to Operating Systems

Fall Semester, 2021

Lab2: Basic C Programming

Submission Deadline: September 9th, 2021 11:59

"The only way to learn a new programming language is by writing programs in it!"
– "Dennis Ritchie".

In this lab, you will do some practices with basic C programming in Linux which you will be using a lot throughout the course. The practices include input, output, conditional statement, loop, function and arrays. However, if you feel you are already an expert in C programming language, you can directly go to the **Exercises** section. Please be aware that only exercies are scored but you may want to practice with other tasks.

Before you start, remember to commit and push your previous lab to your Git repository. Then, try to pull the new lab:

```
$ cd Introduction2OS/labs
$ git pull upstream main
$ cd lab2
```

Please remember these steps for the future labs.

Figure 1: (Picture taken from the Internet)

1 Compiling a C program into an executable file output

Suppose that you have your code in a single file *example.c* and that you want to run your code, type the following command in your terminal:

```
$ gcc example.c -o output
```

This command complies *example.c* into executable file *output*. You can name it whatever you want. Then, type:

```
$ ./output
```

to run the executable file *output* in the current directory. To get you started, we will be going over through creation, compilation, and execution of some simple examples.

Note: There are several ways to compile a C program. You can do whatever it works for you.

2 Output

In C programming language, function `printf` is used to produce output in the form of characters, string, float, integer that are printed on your terminal. This function is in the library ***stdio.h***. Therefore, to use `printf`, you have to add the header file ***stdio.h*** in your code. This is done by preprocessor directive `include` as follows:

```
#include <stdio.h>
```

To generate a newline, use use "\n' in **printf** statement. Now, lets warm up with a simple Greeting program on C. The following example requires you to print out the following output in the terminal:

```
Hello, YourName!
Welcome to C programming.
```

To do that, create the *welcome.c* file by uisng your prefereed editors such as nano, vi, vim, gedit, emacs

```
#include <stdio.h>
int main() {
    printf("Hello, YourName!\n"); //replace YourName with your name
    printf("Welcome to C programming.\n");
    return 0;
}
```

You can now compile your code by using the following command

```
$ gcc welcome.c -o welcome
```

To run your program, type

```
$ ./welcome
```

3 Input

Function `scanf` is used to get input from terminal. The function is also from library *`stdio.h`*. The `scanf` function takes a format string followed by references to where the input should be stored. Remember & in front of variable. This means that the variable is passed as *reference* to `scanf`.

Lets do a simple practice with `scanf`. The following is the code to calculate the summation of two integers input from terminal, however, it is not completed. Use any editor to make your own `.c` and complete the code. Then, run the code.

Sample output:

```
Enter one integer:
Enter another integer:
Sum of the two integers is:
```

Complete the code:

```
#include <stdio.h>
int main() {
       int A, B; // A and B to get input from terminal
       int Sum = 0; // Sum is used to store the sumation of A and B

       printf("Enter one interger:");
       scanf("%d", &A); // d is used for integer
       printf("You have entered %d\n", A);

       printf("Enter another integer:");
       scanf("%d", &B);
       printf("You have entered %d\n", B);

       // Add your own code here

       // End of your code
       printf("Sum of the two integers is %d\n", Sum);
       return 0;
}
```

4 Conditionals

`if-else` statement is used to make a program behave differently depending on the program status or user input. The following code asks for two integers and prints out the their maximum value:

```
#include <stdio.h>
#include <math.h>
int main() {
     int A, B;
     int Max;

     printf("Enter one interger:");
     scanf("%d", &A);
     printf("Enter another integer:");
     scanf("%d", &B);
     if (A < B)
         Max = B;
     else
         Max = A;
     printf("Max is %d ", Max);
}
```

Save the above code in a `.c` file and run the code.

5 Loops

Loops are used to execute a statement or a number of statements multiple times as long as the loop condition is satisfied. The following code calcuates the number of the first 100 integer numbers by using `for` and prints out the result to the terminal:

```
#include<stdio.h>
#include<stdlib.h>

int main(){
      int i;
      int sum = 0;
      for (i = 0; i<100; i++)
             sum+= i;

      printf("Sum is %d\n",sum);
}
```

- Save the above code in a `.c` file and run the code.
- Do the same with `while` loop.

6 Functions

Functions are a great way to make your code reusable, improve the structure of the code, and isolate error. The following is a piece of code to calculate the summation of two integer numbers using functions.

```
#include <stdio.h>
#include <math.h>
int Calculate_Sum(int A, int B) {
     return (A + B);
}
int main() {
    int A, B;
    printf("Enter one interger:");
    scanf("%d", &A);
    printf("Enter another integer:");
    scanf("%d", &B);
    printf("Sum is %d\n", Calculate_Sum(A,B));
}
```

Save the above code in a `.c` file and run the code.

7 Arrays

An array is the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. ... By using array, we can access the elements easily by indices. In this section, we will look at some basic operations on arrays.
You can access elements of an array by indices. Remember that:

- The first index in an array is 0, not 1.
- If the size of an array is N, to access the last element, use index (N-1).

Now, let's do some simple practices with Array

7.1 Task 1

Write a function that asks user to input 10 integers and stores them in an array. Then, print all elements of the array on your terminal.

7.2 Task 2

The following code reads the content of file *Array.txt* which includes 10 integers and put these integers into an array numberArray.

- Print each element of the array numberArray on the terminal.
- Calculate the summation of all elements in numberArray.
- Print the summation on the terminal.

```
#include <stdio.h>
#include <stdlib.h>

int main() {

    FILE* myFile;
    myFile = fopen("Array.txt", "r");

    //read file into array
    int numberArray[10];
    int i;

    if (myFile == NULL) {
        printf("Error Reading File\n");
        exit(0);
    }
    for (i = 0; i <10; i++) {
       fscanf(myFile, "%d,", &numberArray[i]);
   }

    //Start of your code
    // Print numberArray[i] on the screen
    // Calculate the sum of numberArray[]
    // Print the sum

    //End of your code

    fclose(myFile);

    return 0;
}
```

8 Exercises

8.1 Exercise 1 (30 points)

Write a C program that calculates the product of two integer numbers using function.

8.2 Exercise 2 (30 points)

A Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence.

```
F(0) = 0;
F(1) = 1;
F(n) = F(n-1) + F(n-2);
```

Write a C programming to:

- Calculate F(50). **(15 points).**
- Calculate the summation of the first 50 terms in the Fibonacci series. **(15 points).**

8.3 Exercise 3 (60 points)

In an input file which includes all integer numbers, the first element indicates the total number of elements in the file (excluding itself).

- Write a C program to calculate the summation of all elements in *testcase0.txt* except the first element and print the result into screen. **(50 points).**
- Run your above code with *testcase1.txt*. What would you see and what would be your explaination for the output? **(10 points).**

9 What To Submit

Complete the exercises in this lab. Put your codes inside the lab2 directory of your repository. Run `git add` and `git status` to ensure the file has been added and commit the changes by running `git commit -m "Your Commit Message"`. Finally, submit your files to GitHub by running `git push`. Check the GitHub website to make sure that all files have been submitted.