Lab 0

Compiling your first C program in the Linux Environment

Objective:

This lab is intended to provide an introduction to Linux. The objective of this lab is to familiarise students with the Linux command-line environment and develop their skills in shell scripting. This lab serves as a platform for the subsequent labs related to programming in C. The lab consists of :

• Perform basic system operations such as text editing and file management in the Linux environment.

• Introduces the steps for compiling a C program.

Recommended Systems:

- Ubuntu (Should be available in all CSE Lab 3 computers. Use password 1234 if/when needed.)

References:

- Unix concepts and applications, Fourth Edition, Sumitabha Das, TMH.
- Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India.
- Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill.

1. Getting Used to Linux

Linux commands allow you to interact with Linux reliably via a **Terminal**. Typically, a command consists of the command name itself, which specifies the action to be performed (e.g., `ls`, `mkdir`, `cp`). This is often followed by optional arguments or flags, which modify the behavior of the command (e.g., `-l` for a detailed list, `-a` to show all files). Finally, the command usually includes a target, which indicates the file, directory, or other object the command should operate on (e.g., `filename`, `directoryname`). (Note: folders are called *directories*.)
Tip: You can use up/down arrows to move between past commands

Useful Linux Commands:

- Create a new **directory** (i.e a folder): `mkdir progs`
- Go to the progs directory: `cd progs`
- Go to the parent directory: `cd ../`
- Go back to the progs directory: `cd progs`
- List all files in a directory: `ls -a`
- List files with detailed information: `ls -l`
- View a file's content: `cat filename`
- Copy a file to another: `cp file1.c file2.c`
- Copy a file to a directory: `cp file1.c progs/file3.c`
- Move a file to another: `mv file1.c file2.c`
- Move a file to a directory: `mv file1.c progs/file3.c`
- Delete a file: `rm filename`
- Delete an empty directory: `rmdir directoryname`
- Display the current working directory: `pwd`
- Clear the terminal screen: `clear`
- Search for a file: `find . -name "filename"`
- Get help for a command: `man commandname`

2. Compiling Your First C Program

- Click on the terminal icon to open a shell (command prompt).
- The file `hello_world.c` has been provided for your reference. Download it and save it in your working folder.
- Open the program using `gedit` text editor: `$ gedit hello_world.c`

(Note: The `$` symbol represents the command prompt and should not be typed.)

- Write/Edit your program in the editor (if needed) and save it.
- Go to the shell and compile your program: `$ gcc hello_world.c -o hello_world.out`
- This tells the **gcc** C-compiler to **compile** the C-code present in the file `hello_world.c` and create an **executable** file called `hello_world.out`
- If compilation is successful, an executable called `hello_world.out` will be created. If compilation fails, our friend gcc will point out **compilation errors**. Go back to `gedit hello_world.c` and fix those!
- The `hello_world.out` file contains **machine code** that your computer can understand (machine code generated on one computer this way might not directly work on another.)
- Run your program: `$ ./hello_world.out` ("`./`" means the current directory: to run an executable, remember for now, that you need to provide either absolute path or a relative path from the current directory)
- Continue your edit-compile-debug-run-debug-print work as necessary.

FIRST C PROGRAM

```c
#include <stdio.h>

int main(){
    printf("Hello World from my first computer program!\n");
    return 0;
}
```

- Go to the shell and compile your program: `$ gcc hello_world.c`

2. Writing Your Second C Program

`$ gedit second.c`

Write the following code and compile. You can also compile with command `$ gcc second.c` which produces `a.out` and run it with command `$ ./a.out`

Second C program

```
#include <stdio.h>

int main(){
      printf("So excited to write my second C program!\n");
      int a;
      a = 4;
      printf("Value of a is %d\n", a);
      return 0;
}
```

After Second C Program

Make a few mistakes and look at the compiler errors/warnings. Write statements like `printf("Value of a is %d\n", A); printf("Value of a is %%d\n", a); print f("Value of a is %d\n", a); printf("Value of a is %d\n, a); printf("Value of a is %\n", a); printf("Value of a is %d\n", a)` and so on. What do you observe?