# Operating Systems
# ( Lab 4 )

**Date:** Tuesday , 13/3/2018

## Contents and topics to be covered:

- ***Programming Under Linux***
  1. *GCC and G++ compilers*
  2. *C/C++ Programming Under Linux*
  3. *Java Programming Under Linux*
  4. *Sample Hello World program*
     - *Using C language*
     - *Using C++ language*
     - *Using Java language*

---

**Commands you may need:**

```
Sudo apt install gcc

Sudo apt install g++

sudo apt install default-jdk

sudo snap install --classic eclipse

sudo snap install netbeans --classic

su

sudo -i

passwd

su - user

sudo adduser user-name
```

- ***GCC and G++ compilers***

  - "GCC" is a common shorthand term for the GNU Compiler Collection. This is both the most general name for the compiler, and the name used when the emphasis is on compiling C programs (as the abbreviation formerly stood for "GNU C Compiler").

  - When referring to C++ compilation, it is usual to call the compiler "G++". Since there is only one compiler, it is also accurate to call it "GCC" no matter what the language context; however, the term "G++" is more useful when the emphasis is on compiling C++ programs.

    How do I use gcc, g++, and gdb?

The C compiler on eniac is gcc. Its C++ counterpart is g++.

To compile a C or C++ program:

```
% gcc file.c
```

or

```
% g++ file.c
```

This compiles `file.c` into an executable binary named `a.out`.

Here are a few options to gcc and g++:

**-o *outputfile***

To specify the name of the output file. The executable will be named `a.out` unless you use this option.

**-g**

To compile with debugging flags, for use with gdb.

**-L *dir***

To specify directories for the linker to search for the library files.

**-l *library***

This specifies a library to link with.

**-I *dir***

This specifies directories for the compile to search for when looking for include files.

## *Example 1: Compiling a simple C program*

Below is the Hello-world C program `hello.c`:

```
1 // hello.c
2 #include <stdio.h>
3
4 int main() {
5     printf("Hello, world!\n");
6     return 0;
7 }
```

To compile the `hello.c`:

```
> gcc hello.c
  // Compile and link source file hello.c into executable a.exe
```

The default output executable is called "`a.exe`". For Cygwin under CMD shell, use `gcc-4` or `gcc-3`, instead of `gcc`.

To run the program:

```
// Under Bash or Bourne Shell - include the current path (./)
$ ./a
```

To specify the output filename, use `-o` option:

```
$ gcc -o hello.exe hello.c
  // Compile and link source file hello.c into executable hello.exe
$ ./hello
  // Execute hello.exe under Bash or Bourne shell, specifying the current
path (./)
```

### *Example 1: Compiling a simple C++ program*

```
1 // hello.cpp
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6    cout << "Hello, world!" << endl;
```

```
7    return 0;
8 }
```

You need to use g++ to compile C++ program, as follows. We use the `-o` option to specify the output file name.

```
$ g++ -o hello.exe hello.cpp
   // Compile and link source hello.cpp into executable hello.exe
$ ./hello
   // Execute under Bash or Bourne shell, specifying the current path (./)
```

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

Compile it by typing the `javac` command below:

```
$ javac HelloWorld.java
```

Execute it by typing the `java` command below:

```
$ java HelloWorld
```