

Get Started with C++ In Linux

Contents

Get Started with C++ In Linux	1
Activity 1: Hello World with C++	1
Activity 2: If Decision Statements with C++	2
Activity 3: The While Loop with C++	5
Activity 4: The Do While Loop with C++	6
Activity 5: The For Loop with C++	7
Activity 6: Find Syntax Errors with GCC	9
Assignment Submission.....	10

Time required: 90 minutes

NOTE: Please read the book as you are going through these activities to gain an understanding of programming concepts.

Activity 1: Hello World with C++

Objective: Explore some of the programming constructs of the C++ programming language. It is traditional to start out with a simple Hello World program in whatever programming language you are learning. This program confirms that your programming environment is functional.

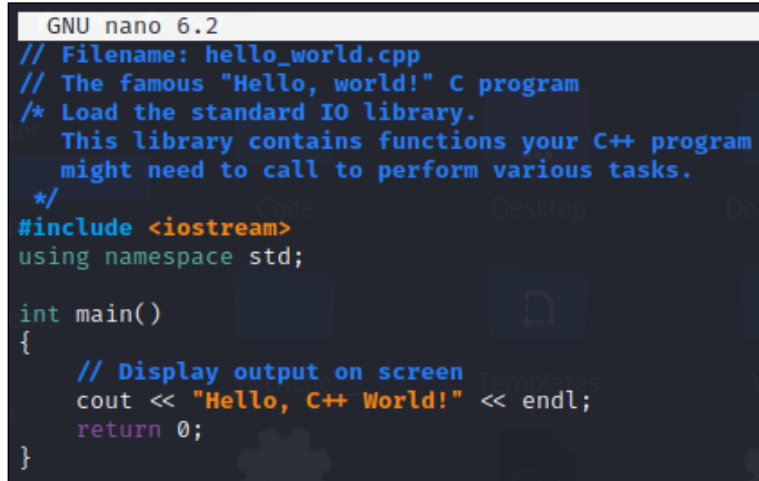
We will be using the free open-source **GNU Compiler Collection**, commonly known as GCC, to compile our C++ programs. GCC is a set of compilers and development tools available for Linux, Windows, various BSDs, and a wide assortment of other operating systems. It includes support primarily for C and C++ and includes Objective-C, Ada, Go, Fortran, and D.

We will be writing our code in C++, we will use the g++ compiler.

Description: Write and compile the traditional Hello World program.

1. Boot your computer into **Kali Linux**.
2. At the shell prompt, type **nano hello_world.cpp** Press Enter to use the nano editor.

3. Add the following code.



```
GNU nano 6.2
// Filename: hello_world.cpp
// The famous "Hello, world!" C program
/* Load the standard IO library.
   This library contains functions your C++ program
   might need to call to perform various tasks.
*/
#include <iostream>
using namespace std;

int main()
{
    // Display output on screen
    cout << "Hello, C++ World!" << endl;
    return 0;
}
```

4. Save the program file: Press **CTRL S**
5. Exit nano: **CTRL X**
6. To compile your program type: **g++ -o hello_world hello_world.cpp**
 - a. **G++**: Open-source C++ compiler
 - b. The **-o** switch tells the compiler to create an output file called **hello_world**
 - c. Source file: **hello_world.cpp**
7. If there isn't any output, the program compiled successfully.
8. To run your program: **./hello_world**

Example run:



```
(user@kali)-[~]
└─$ g++ -o hello_world hello_world.cpp

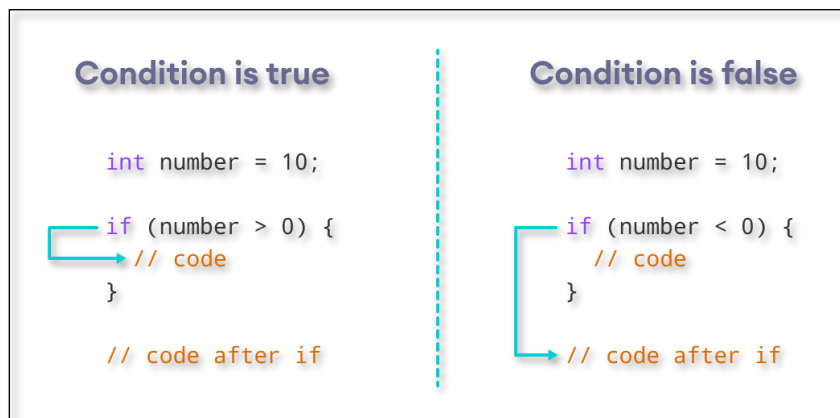
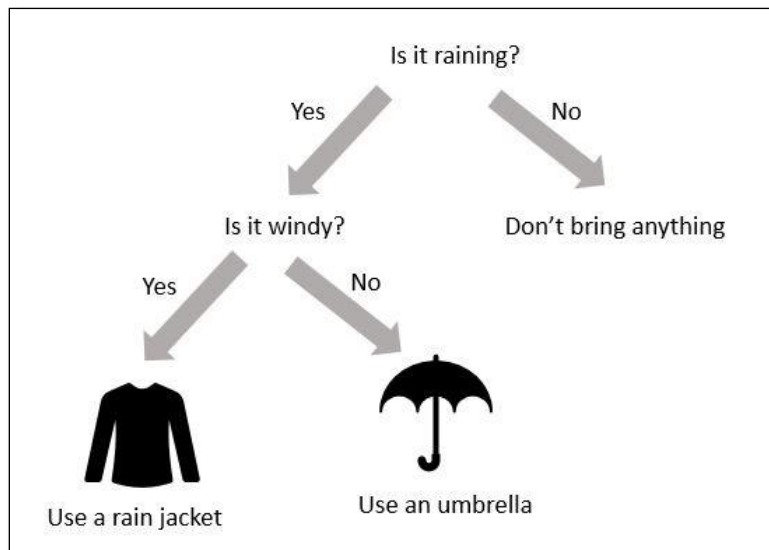
(user@kali)-[~]
└─$ ./hello_world
Hello, C++ World!

(user@kali)-[~]
└─$
```

Activity 2: If Decision Statements with C++

Objective: Learn about decision making in C++.

If statements are based on Boolean conditional expressions. There is a test to determine whether the condition is true or false.



The following code compares two numbers to determine which is larger. Change the numbers to see how the code works.

1. Boot your computer into **Kali Linux**.
2. At the shell prompt, type **nano if_decision.cpp** Press Enter to use the nano editor.

```

1  // Filename: if_decision.cpp
2  // Demonstration of if decision making
3  #include <iostream>
4
5  int main()
6  {
7      // Initialize integer variables
8      int x{0};
9      int y{10};
10
11     std::cout << "x is " << x << " and y is " << y << std::endl;
12
13     // Compare x to y, if x is greater than y, condition is true
14     if (x > y)
15     {
16         std::cout << "x is greater than y\n";
17     }
18
19     // Compare x to y, if x less than or equal to y, condition is false
20     else if (x <= y)
21     {
22         std::cout << "x is not greater than y\n";
23     }
24     return 0;
25 }

```

1. Save the program file: Press **CTRL S**
2. Exit nano: **CTRL X**
3. To compile your program: **g++ -o if_decision.cpp**
4. If there isn't any output, the program compiled successfully.

To run your program: **./if_decision**

Example run:

```

(user@kali)-[~]
$ g++ -o if_decision if_decision.cpp

(user@kali)-[~]
$ ./if_decision
x is 0 and y is 10
x is not greater than y

(user@kali)-[~]
$

```

Activity 3: The While Loop with C++

Objective: Learn about loops in C++

Description: There are three loop types in most programming languages. While, Do While, and For. We will explore each of these with the C++ programming language.

5. Boot your computer into **Kali Linux**.
6. At the shell prompt, type **nano while_loop.cpp** Press Enter to use the nano editor.

```

1  // Filename: while_loop.cpp
2  #include <iostream>
3
4  int main()
5  {
6      // Initialize (assign a value to) the counter variable
7      int counter{1};
8      // Do what's inside the braces until false, greater than 10
9      while (counter <= 10)
10     {
11         std::cout << "Counter is equal to " << counter << std::endl;
12         // Increment counter by 1;
13         ++counter;
14     }
15     return 0;
16 }

```

The While loop is a pretest loop. If the counter is greater than 10, the loop will never run. In this case, when the counter variable is greater than 10, the while loop stops processing. This causes cout to display 10 lines of output before stopping.

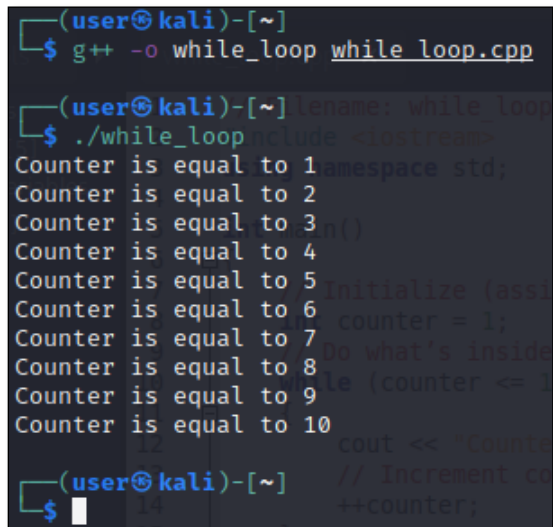
7. Save the program file: Press **CTRL S**
8. Exit nano: **CTRL X**

9. To compile your program: **g++ -o while_loop while_loop.cpp**

10. If there isn't any output, the program compiled successfully.

To run your program: **./while_loop**

Example run:



```
(user@kali)-[~]
$ g++ -o while_loop while_loop.cpp

(user@kali)-[~]
$ ./while_loop
Counter is equal to 1
Counter is equal to 2
Counter is equal to 3
Counter is equal to 4
Counter is equal to 5
Counter is equal to 6
Counter is equal to 7
Counter is equal to 8
Counter is equal to 9
Counter is equal to 10
```

Activity 4: The Do While Loop with C++

Objective: Learn about loops in C++

Description: There are three loop types in most programming languages. While, Do While, and For. We will explore each of these with the C++ programming language.

1. Boot your computer into **Kali Linux**.
2. At the shell prompt, type **nano do_while_loop.cpp** Press Enter to use the nano editor.

```

1  // Filename: do_while_loop.cpp
2  #include <iostream>
3
4  int main()
5  {
6      // Initialize (assign a value to) //the counter variable
7      int counter{1};
8      // Do what's inside the braces until false
9      do
10     {
11         std::cout << "Counter is equal to " << counter << std::endl;
12         // Increment counter by 1;
13         ++counter;
14     } while (counter <= 10);
15     return 0;
16 }

```

The Do While loop is a post test loop. The loop runs at least once before the counter is tested. When the counter variable is greater than 10, the while loop stops processing, which causes cout to display 10 lines of output before stopping.

Example run:

```

(user@kali)-[~]
$ g++ -o do_while_loop do_while_loop.cpp

(user@kali)-[~]
$ ./do_while_loop
Counter is equal to 1
Counter is equal to 2
Counter is equal to 3
Counter is equal to 4
Counter is equal to 5
Counter is equal to 6
Counter is equal to 7
Counter is equal to 8
Counter is equal to 9
Counter is equal to 10

```

Activity 5: The For Loop with C++

The last loop type in C is the for loop, one of C's most interesting pieces of code. A for loop starts with the keyword **for** followed by three items in starting and ending round brackets (also called parentheses).

1. The first item inside the brackets initializes the variable that the for loop will use.
int counter = 1
2. The second item defines a test that if false, causes the for loop to exit.
counter <= 10
3. The third item is the counter.
counter++
4. After the for statement, you place any code you want to execute in starting and ending curly brackets (also called braces). The code inside the curly brackets will be executed for each iteration of the for loop.

```
1 // Filename: for_loop.cpp
2 #include <iostream>
3
4 int main()
5 {
6     // Do what's inside the braces until false
7     for (int counter{1}; counter <= 10; counter++)
8     {
9         // Print even numbers using % modulus
10        // If the remainder is 0, the number is even
11        if (counter % 2 == 0)
12        {
13            std::cout << "Counter is equal to " << counter << std::endl;
14        }
15    }
16    return 0;
17 }
```

Example run:

```
(user@kali)-[~]
$ g++ -o for_loop for_loop.cpp

(user@kali)-[~]
$ ./for_loop
Counter is equal to 2
Counter is equal to 4
Counter is equal to 6
Counter is equal to 8
Counter is equal to 10

(user@kali)-[~]
$
```


Activity 6: Find Syntax Errors with GCC

Objective: Learn how to use the GNU GCC compiler included with most *nix operating systems.

Description: In the past, programmers had to read through their code line by line before submitting the job to the mainframe CPU. The job included all the commands the CPU would execute. If a program was full of errors, the mainframe operator notified the programmer, who had to go through the code again and fix the errors.

With today's compilers, you can write a program, compile it, and test it yourself. If the compiler finds errors, it usually indicates what they are so that you can correct the code and compile the program again.

In this activity, you create a C program that contains errors and try to compile the program. After seeing the errors generated, you correct the program and then recompile it until you get it right.

3. Boot your computer into **Kali Linux**.
4. At the shell prompt, type **nano syntax.cpp** Press Enter to use the nano editor.

Type the following code, pressing Enter after each line:

```
1 // Filename: syntax.cpp
2 #include <iostream>
3
4 int main()
5 {
6     int age;
7     std::cout << "Enter your age: "
8     std::cin >> age;
9     if (age > 0)
10    {
11        std::cout << "You are " << age << " years old" << std::endl;
12    }
13    return 0;
14 }
```

5. Press **CTRL S** to save. Press **CTRL X** to exit
6. To compile the program, type **gcc -o syntax syntax.cpp** and press **Enter**. The **-o** switch tells the compiler to create an output file called syntax. The compiler returns an error (or several errors) like what is shown below.

```
(user@kali)-[~]
$ g++ -o syntax syntax.cpp
syntax.cpp: In function 'int main()':
syntax.cpp:8:31: error: expected ';' before 'cin'
  8 |     cout << "Enter your age: "
    |                               ^
  9 |     cin >> age;
    |     ~~~
(user@kali)-[~]
$
```

7. The error varies depending on the compiler version you use. In any event, you should be warned that there was a syntax error before `cin` because there was no semicolon after the `int age` statement.
8. Correct the missing semicolon error by typing: **nano syntax.cpp**
9. Add a semicolon to the end of the line containing the variable declaration `int age`.
10. Save and exit the program.
11. Compile the program again by typing **gcc -o syntax syntax.cpp** and pressing Enter. (You can also use the up arrow key to return to previous commands.)
12. If you entered everything correctly, you should be at the shell prompt.
13. To run the program, type **./syntax** and press Enter.

Example run:

```
Enter your age: 24
You are 24 years old
```

Assignment Submission

1. Attach all files.
2. Attach a screenshot of each successful program run in Kali.
3. Submit the assignment in Blackboard.