Linux Get Started with C++

Contents

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.

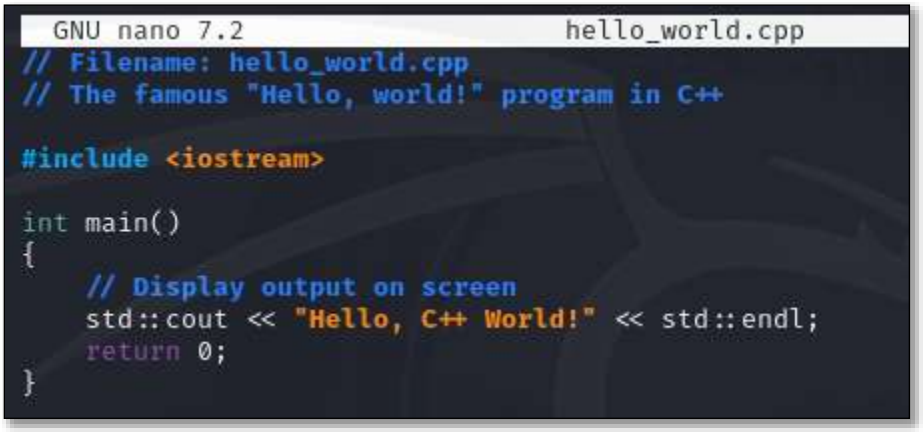
We will be writing our code in C++, we will use the g++ compiler.

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 to your Hello World program.

NOTE: C++ doesn't care how you indent like Python does. C++ aligns what code belongs to what by the curly braces. A single tab is a good way to ident and align the code.



Explanation:

- **#include <iostream>** - Load the standard IO library, iostream This library contains functions your C++ program might need to call to perform various tasks.
    - std:: is the standard IO library
    - cout → output to console
    - endl → end line, move to next line
- **int main()** - Entry point for all C++ programs
- **return 0**; **-** This line is at the end of all C++ programs. It returns 0 to the operation system, meaning the program executed successfully

Compile and run the program:

4. Save the program file: Press **CTRL S**
5. Exit nano: **CTRL X**
6. To compile your program type: **./**
    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.

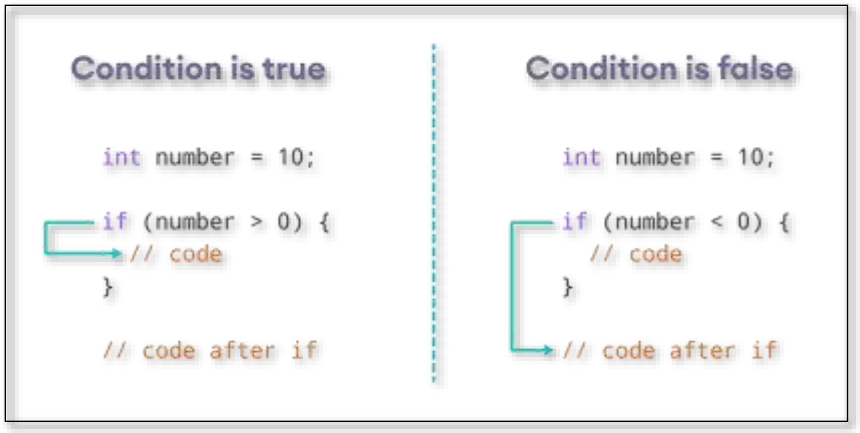
8. To run your program: **./hello_world**

Example run:

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 wither 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.

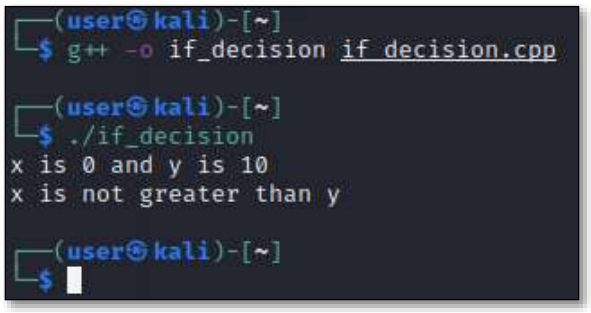
1. Boot your computer into **Kali Linux**.
2. At the shell prompt, type **nano if_decision.cpp** Press Enter to use the nano editor.

```cpp
// Filename: if_decision.cpp
// Demonstration of if decision making
#include <iostream>

int main()
{
    // Initialize integer variables
    int x{0};
    int y{10};

    std::cout << "x is " << x << " and y is " << y << std::endl;

    // Compare x to y, if x is greater than y, condition is true
    if (x > y)
    {
        std::cout << "x is greater than y\n";
    }

    // Compare x to y, if x less than or equal to y, condition is false
    else if (x <= y)
    {
        std::cout << "x is not greater than y\n";
    }
    return 0;
}
```

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

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

Example run:



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.

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

```cpp
// Filename: while_loop.cpp
#include <iostream>

int main()
{
    // Initialize (assign a value to) the counter variable
    int counter{1};
    // Do what's inside the braces until false, greater than 10
    while (counter <= 10)
    {
        std::cout << "Counter is equal to " << counter << std::endl;
        // Increment counter by 1;
        ++counter;
    }
    return 0;
}
```

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.

3. Save the program file: Press **CTRL S**
4. Exit nano: **CTRL X**
5. To compile your program: **g++ -o while_loop while_loop.cpp**
6. 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

┌──(user㉿kali)-[~]
└─$
```

Activity 4: The Do While Loop with C++

Objective: Learn about loops in C++

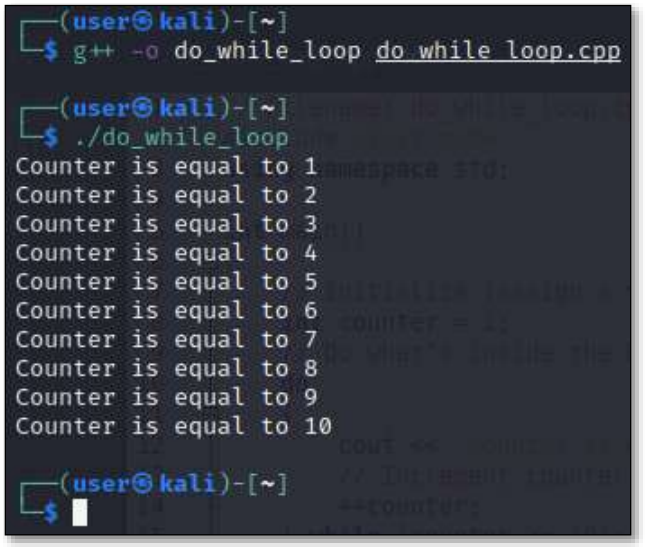
Description: There are three loop types in most programming languages. While, Do While, and For. We will explore each of these with the C++ programming language.

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.

```cpp
// Filename: do_while_loop.cpp
#include <iostream>

int main()
{
    // Initialize (assign a value to) //the counter variable
    int counter{1};
    // Do what's inside the braces until false
    do
    {
        std::cout << "Counter is equal to " << counter << std::endl;
        // Increment counter by 1;
        ++counter;
    } while (counter <= 10);
    return 0;
}
```

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:



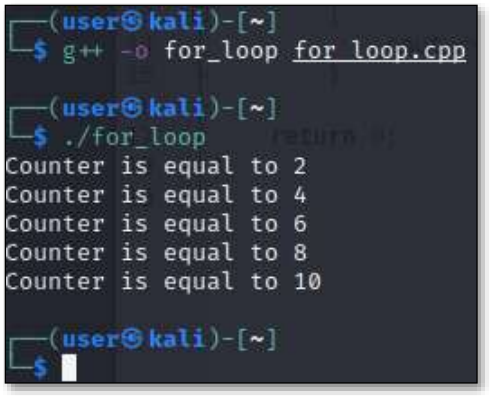
Activity 5: The For Loop with C++

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.

```cpp
// Filename: for_loop.cpp
#include <iostream>

int main()
{
    // Do what's inside the braces until false
    for (int counter{1}; counter <= 10; counter++)
    {
        // Print even numbers using % modulus
        // If the remainder is 0, the number is even
        if (counter % 2 == 0)
        {
            std::cout << "Counter is equal to " << counter << std::endl;
        }
    }
    return 0;
}
```

Example run:



Assignment Submission

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