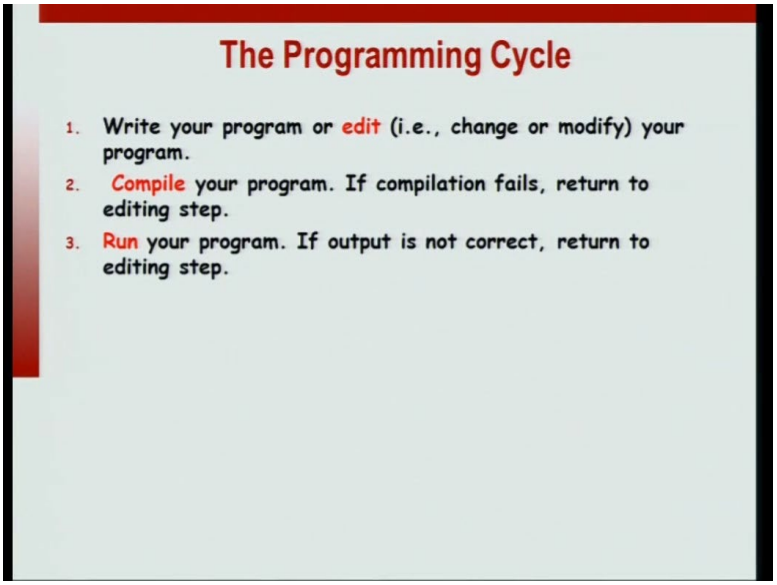
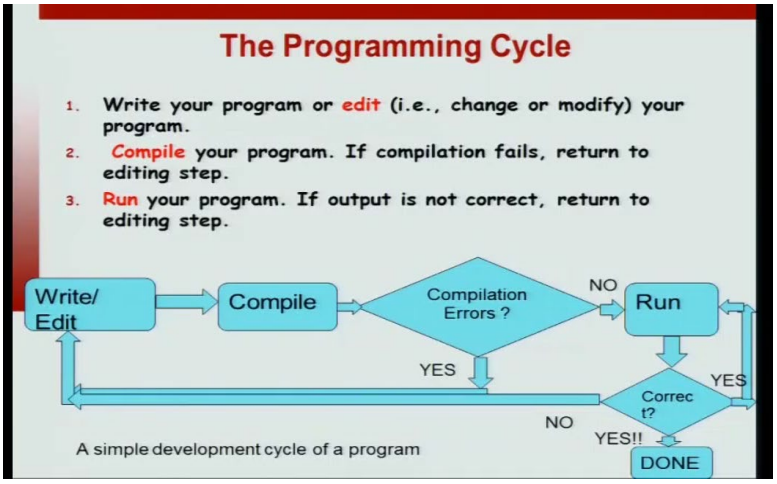
Introduction to Programming in C
Department of Computer Science and Engineering

(Refer Slide Time: 00:22)



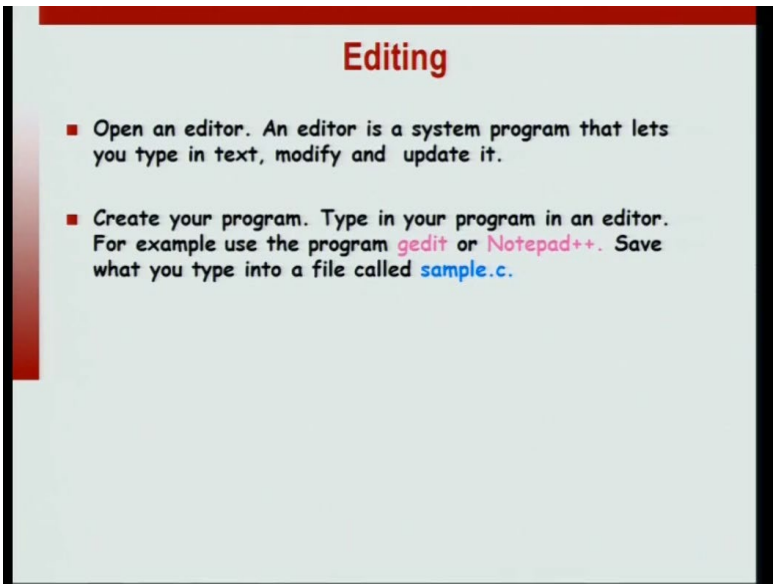
Once we are understood what algorithms are, we will start writing a few simple programs in the C programming language. Before we begin, we will give a brief introduction to the process of programming. When you are programming, you follow typically, what is known as the programming cycle and this contains three parts. One is the process where you write the program or edit the program, and after you are done editing the program, you save it and then you compile your program. If your compilation succeeds, you are ready to run the program. If your compilation fails, then you return to the editing step and correct the errors and compile again. Once compilation process succeeds, then you can run the program and check whether the output is correct. If the output is correct, you are done; if not you go back to the edit process.

(Refer Slide Time: 00:59)



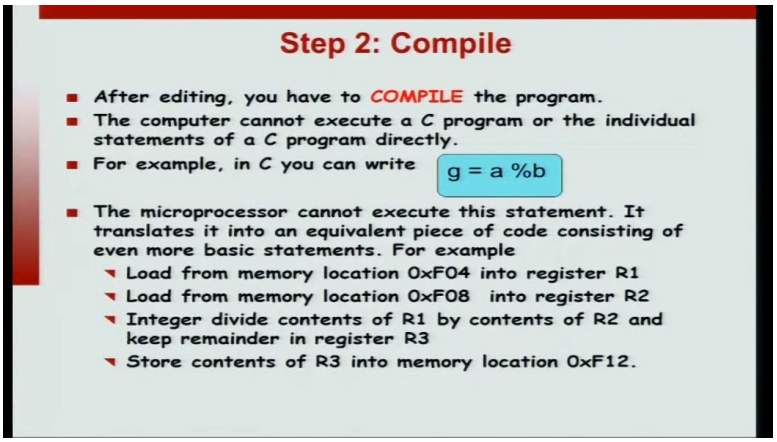
So, this is why it is known us the edit, compile, run cycle. So, you edit the program first, then compile it. If there are compilation errors you go back and edit it again, otherwise you run the program. When you run the program, if the logic is correct, then you are done. If your logic is incorrect, then you go back and make changes to the program, compile it and run it again. So, this is the process, that we have to follow in the, when we program. We look at each of the steps one by one.

(Refer Slide Time: 01:38)



In editing, it is typically done in what is known as in editor. Now, editor is a program that lets you create a text file, make changes to the text file and update the text file, later save it. So, in order to create a program, pick up particular editor of your choice. If you are on Linux, I would recommend a simple editor like G Edit. If you are on windows, there is free editor called Notepadd + +. Be careful that this is not the usual notepad that comes along with the system. Write your code in, in editor of your choice and save it into a file. Let us call it Sample.c.

(Refer Slide Time: 02:25)



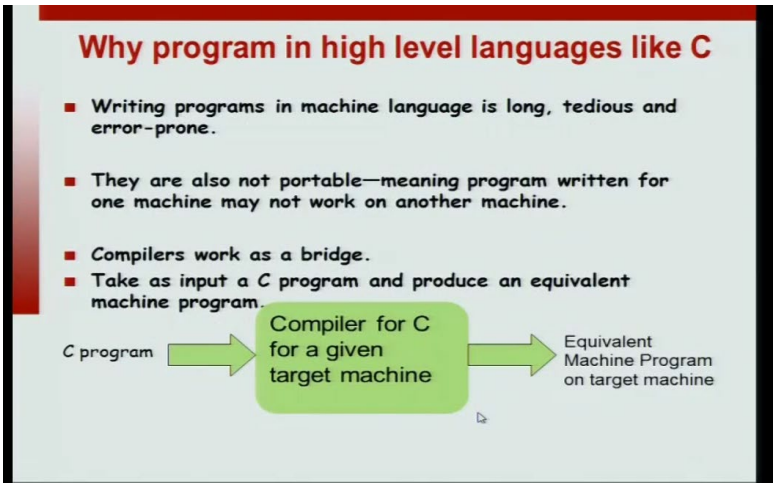
Once your code is saved, you have to compile a program. Now, why do we have to compile a program? Why is this step necessary? The computed does not understand C per say, it cannot execute a C program or the individual statements in a C, in C language correctly. For example, let us say that in C you can write g = a % b. The percentage operation stands for modulo. So, this statement says that you take a % b and assign it to the variable g.

Now, the microprocessor, the processor in the computer cannot execute this statement because it does not understand this C programming language. So, it translates it into an equivalent piece of code consisting of even more basic statements. For example, a, this is just for the purpose of illustration and it is not important that you understand exactly what is going on, but in a statement like g = a % b, can be translated into bunch of statements saying load data from particular memory location into particular register, load the second piece of data from another memory location to the second register, divide the

contents of these two registers, store the reminder in a third register and then finally, take the result and store it into a third memory location.

So, the simple statement that we wrote, g = a % b or g = a % b, becomes a bunch of basic statements, that the microprocessor can understand and then it execute these statements.

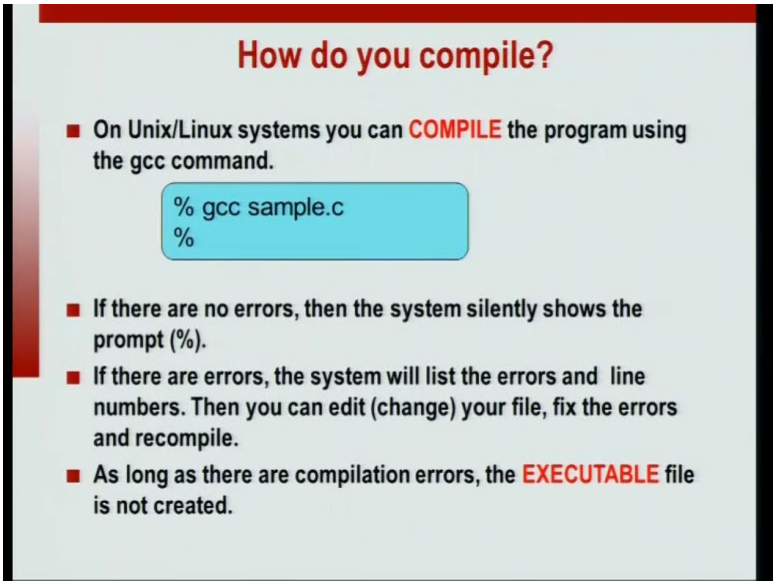
(Refer Slide Time: 04:17)



So, why not program in the microprocessor language or in assembly language? Writing programs in machine language is very tedious. One line in a higher programming language like C translates into multiple lines of machine language. So, writing machine language code is very long and it is very tedious and is particularly prone to errors. Also, they are not portable. If you write machine code for a particular processor, let us say, you are writing the code for an Intel processor and you translate it to an AMD machine, it might not work. Whereas, if you take your C code and compile it in another machine, it will run on the machine.

So, compilers work as a bridge. What they do is, take a high level C programming language and translate it into the equivalent machine code. So, think of them as a translator. So, you, the input is a C program and then you give it to a compiler. The output of the compiler will be the equivalent machine program for whichever machine you want to run it on. So, compiler is a translator, which translates from C to machine code.

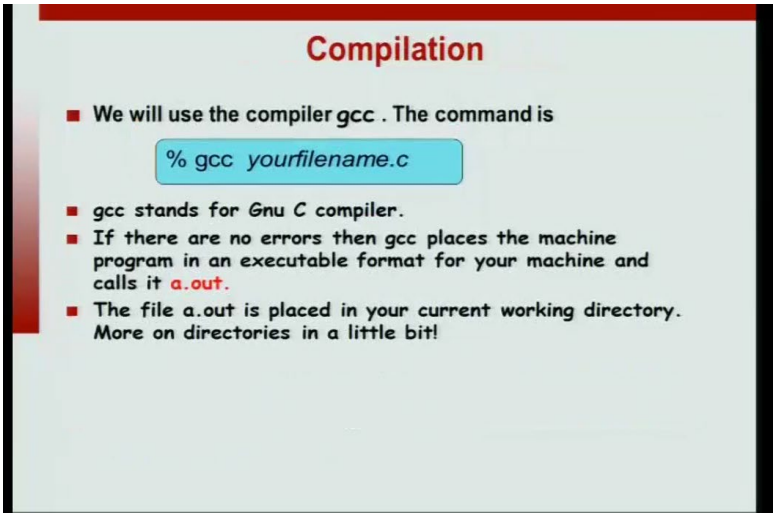
(Refer Slide Time: 05:36)



How do you compile? We have just seen why we bother with compilation and on UNIX system or Linux systems, you can compile the program using the gcc compiler. So, gcc stands for the gnu c compiler. So, for example, if you have edited and saved your file as a Sample.c, you can just type on the comment prompt on the terminal gcc Sample.c.

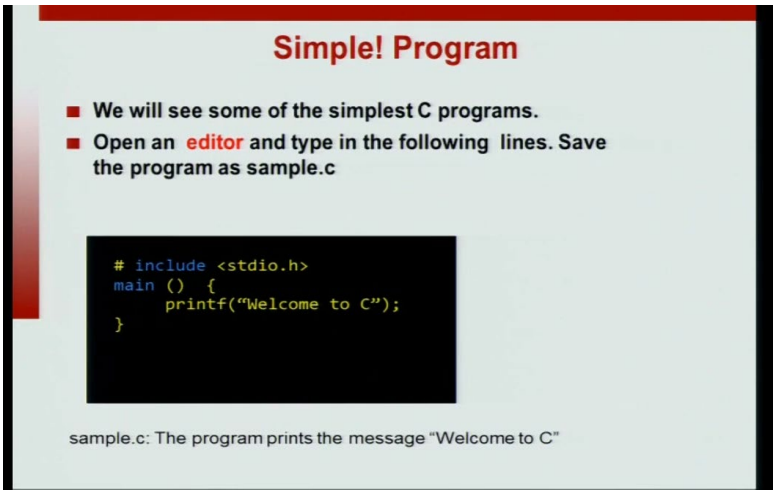
If your code does not have any errors, then the system will silently say, that the compilation is done and it will show you the prompt. If there are errors, the system will list the errors and so, you can go back to the editor, edit you code to correct errors and come back and compile again. As long as there are compilation errors, there will be no executable file created. So, the executable file is the code, is the file that you can finally run. And if there are compilation errors, the compiler will not produce executable code.

(Refer Slide Time: 06:43)



So, name your file as whatever you want, let us call it, yourfilename.c and then gcc yourfilename.c. It will produce the executable file. If you are on Linux, the executable file that it creates is something called a.out. If there are no errors and look at your directory, there will be a new file called a.out in your directory and we will explain the directory structures in another session, ok.

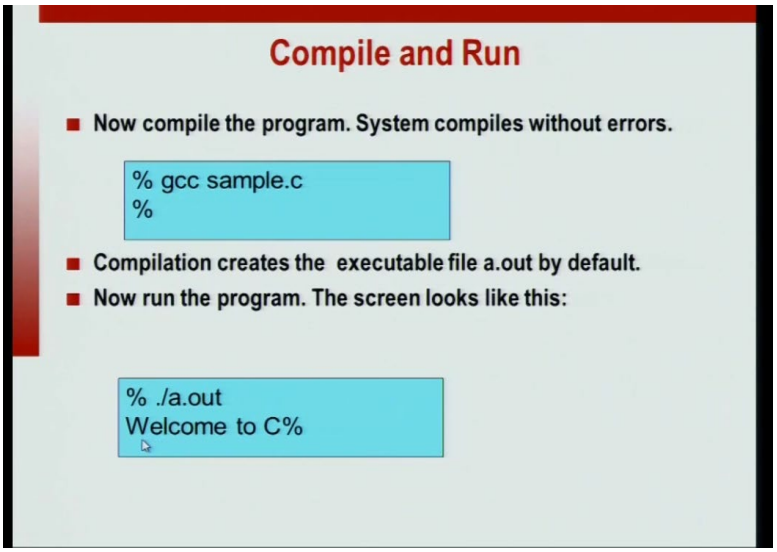
(Refer Slide Time: 07:16)



Let us look at a very simple C program. Open your editor depending on which system you are in. So, let us write a very simple program. It is, it is very short. What it has is, are

three lines of code and some punctuation. This is known as the C syntax. Let us examine this code. What this code does is, it prints a particular massage, which is, welcome to C.

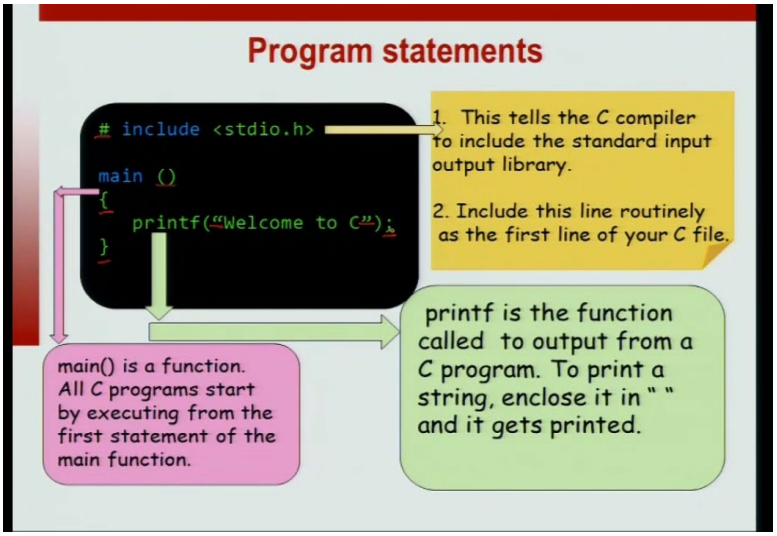
(Refer Slide Time: 07:51)



And it has various components, you type it into an editor as it is, make no punctuation mistakes, syntax errors. Now, if you compile the program and you have typed the program correctly, then a new file called a.out will be created. So, if you type, gcc Sample.c and if there are no errors, it will just say nothing. If there are, if it says something, then there is a compilation error.

Compilation creates an executable a.out and now you can run the program by typing, and this is important, ./a.out. So, this syntax is important, what you type is, ./a.out and then when you run the program it will say, welcome dot, Welcome to C, because that is what the program is supposed to do…

(Refer Slide Time: 08:47)



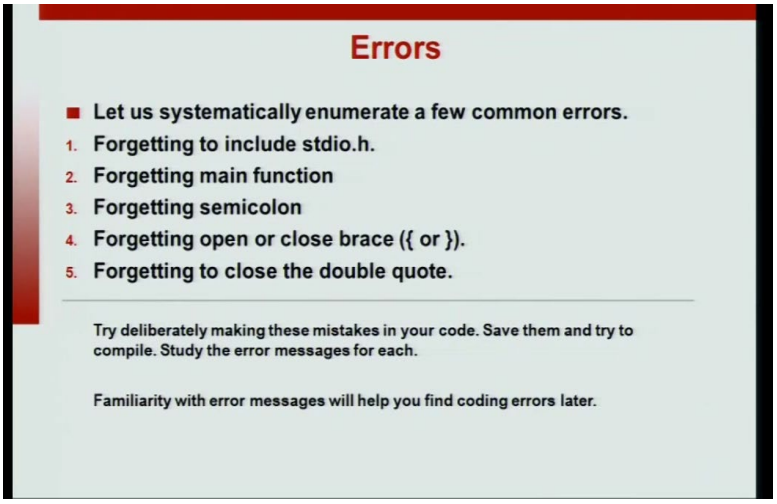
Let us look at the program little more carefully. What are its components? It had three lines, the first line said # include <stdio.h>. So, it has multiple components. One is the first symbol, which is, has the first symbol, which is the hash. Please do not forget to include that. And actually, there is no space between the hash and the first i, so there is no space here. So, # include <stdio.h>. This line is supposed to tell C that please include the standard input-output library. The standard input-output library is what has the print routines, which will print output messages on to the terminal.

So, if you want to have any input output component of your program, then you should include <stdio.h>. Include this line routinely in your, in the first line of your C file because in the course of this class, we will often need scanf and printf statements. So, we will often need input statement and output statements. So, include this by default.

Now, if you look at the second line, we will have a function called main. And again, note the parenthesis here that is also part of the syntax. So, main is supposed to be a function. All C programs start by executing the main function and it starts from first statements of the main function. Now, what dose the main function have? It has a single line, which says printf Welcome to C. So, printf is the function called to output from a C program. So, to print a particular massage you enclose it in double quotes. So, whatever is enclosed in the double codes, will be printed.

So, to repeat again, please note the extra punctuation symbols, which tell you, that these are valid C statements. So, all the underline statements are, all the underline symbols are important. So, in the line printf Welcome to C, this is what is known as a statement in C and statements in C end in a semicolon. So, this semicolon is also important because it tells you, that this is where the statements ends, what typical errors do we have when we code in C.

(Refer Slide Time: 11:34)



Let us systematically enumerate a few common errors that could happen in even a simple program like what we have seen. For example, you could forget to include <stdio.h>. If you do not include the standard input library, then the compiler will give you an error message. You may forget to include the main function, then also you will get some error message. You could forget to include the semicolon in the statement, you could forget to include the braces, the curly braces in main or forget to close the double quote, open or close the double quote in the printf statement. So, these are a few errors that you could make even in a simple code like what we have seen.

We have only three lines, but they could also have errors. I would advise you to try deliberately making these mistakes in your code, try compiling them and study the error messages. Once you are familiar with error massages, this will help you later in your coding, because when you see the error massages you can guess what errors did you possibly make in your code. So, go back to the code and correct it.