

C Programming Lecture 1

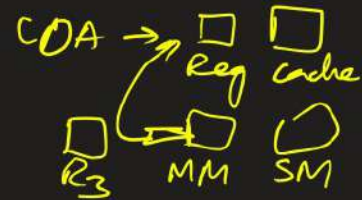
Monday, 3 June 2024 6:59 PM

- **C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- **Dennis Ritchie** is known as the **founder of the C language**.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.
- Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL.



Punch
cards

↓
low-language



Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard }
B	1970	Ken Thompson }
Traditional C	1972	<u>Dennis Ritchie</u> }
K & R C	1978	Kernighan & Dennis Ritchie
<u>ANSI C</u> ✓	1989	<u>ANSI Committee</u>
ANSI/ISO C	1990	ISO Committee
<u>C99</u>	1999	<u>Standardization Committee</u>

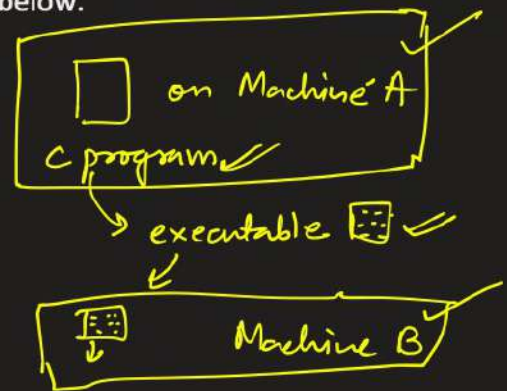
Features of C Language

C is the widely used language. It provides many **features** that are given below.

1. Simple ✓ → *structured, easy to learn*
2. Machine Independent or Portable ✓
3. Mid-level programming language ✓
4. Structured programming language ✓
5. Rich Library ✓
6. Memory Management ✓
7. Fast Speed ✓
8. Pointers ✓
9. Recursion ✓



Java
Python
C++
C - midlevel



I/P → [Fn] processing

I/P → [Fn] → O/P
processing

First C Program

```
↓ ↓
// Hello World Program
#include <stdio.h> ← header file
int
void main() {
    printf("Hello World");
    return 0;
}
```

Processing

← Preprocessing directive

Abstraction

General Overview of a Simple C Program's Structure:

The **general architecture** of a simple **C program** typically consists of several vital components. Below is an outline of the essential elements and their purposes:

- **Header Files:**

The **#include directives** at the beginning of the program are used to include **header files**. **Header files** provide function **prototypes** and **definitions** that allow the C compiler to understand the functions used in the program.

- **Main Function:**

Every **C program** starts with the **main function**. It is the program's entry point, and execution starts from here. ✓
The **main function** has a **return type** of **int**, indicating that it should return an integer value to the operating system upon completion. = void ✓

- **Variable Declarations:** ✓

Before using any variables, you should declare them with their **data types**. This section is typically placed after the **main function's** curly opening brace.

- **Statements and Expressions:** ✓✓

This section contains the *actual instructions* and *logic* of the program. C programs are composed of statements that perform *actions* and *expressions* that compute values.

- **Comments:** ✓✓

Comments are used to provide human-readable explanations within the code. They are not executed and do not affect the program's functionality. In C, comments are denoted by // for *single-line comments* and /* */ for *multi-line comments*.

- **Functions:** ✓✓

C programs can include user-defined functions and blocks of code that perform specific tasks. Functions help modularize the code and make it more organized and manageable.

- **Return Statement:**

Use the return statement to terminate a function and return a value to the caller function. A *return statement* with a value of 0 typically indicates a successful execution in the *main function*, whereas a *non-zero value* indicates an error or unexpected termination.

- **Standard Input/Output:**

C has library functions for reading user input (scanf) and printing output to the console (printf). These functions are found in C programs and are part of the standard I/O library (stdio.h header file). It is essential to include these fundamental features correctly while writing a simple C program to ensure optimal functionality and readability.

GNU → GNU not UNIX GCC → GNU C compiler

Install your distribution's build tools. Most versions of Linux don't come with [GCC](#) already installed. Fortunately, it's easy to install GCC and other required tools (including *make*, G++, and general development libraries) for compiling software on any version of Linux:

- Ubuntu, Debian, & Linux Mint:

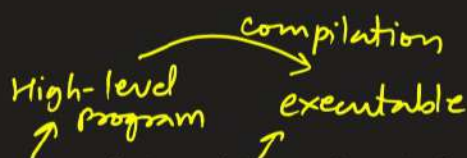
```
sudo apt update
```

```
sudo apt install build-essential
```

```
gcc --version
```

```
wikihow@wikihow-UB: ~  
wikihow@wikihow-UB:~$ sudo apt update ✓  
[sudo] password for wikihow:  
Hit:1 http://in.archive.ubuntu.com/ubuntu kinetic InRelease  
Hit:2 http://in.archive.ubuntu.com/ubuntu kinetic-updates InRelease  
Hit:3 http://in.archive.ubuntu.com/ubuntu kinetic-backports InRelease  
Hit:4 http://security.ubuntu.com/ubuntu kinetic-security InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
5 packages can be upgraded. Run 'apt list --upgradable' to see them.  
wikihow@wikihow-UB:~$ sudo apt install build-essential ✓  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
build-essential is already the newest version (12.9ubuntu3).  
build-essential set to manually installed.  
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.  
wikihow@wikihow-UB:~$ gcc --version ✓  
gcc (Ubuntu 12.2.0-8ubuntu1) 12.2.0  
Copyright (C) 2022 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
wikihow@wikihow-UB:~$  
MinGW GCC ← Windows-
```

Compilation process in C

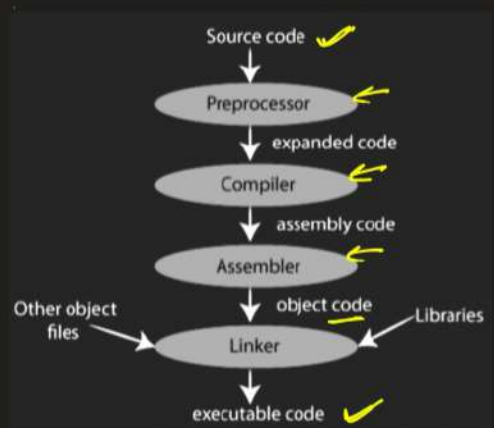


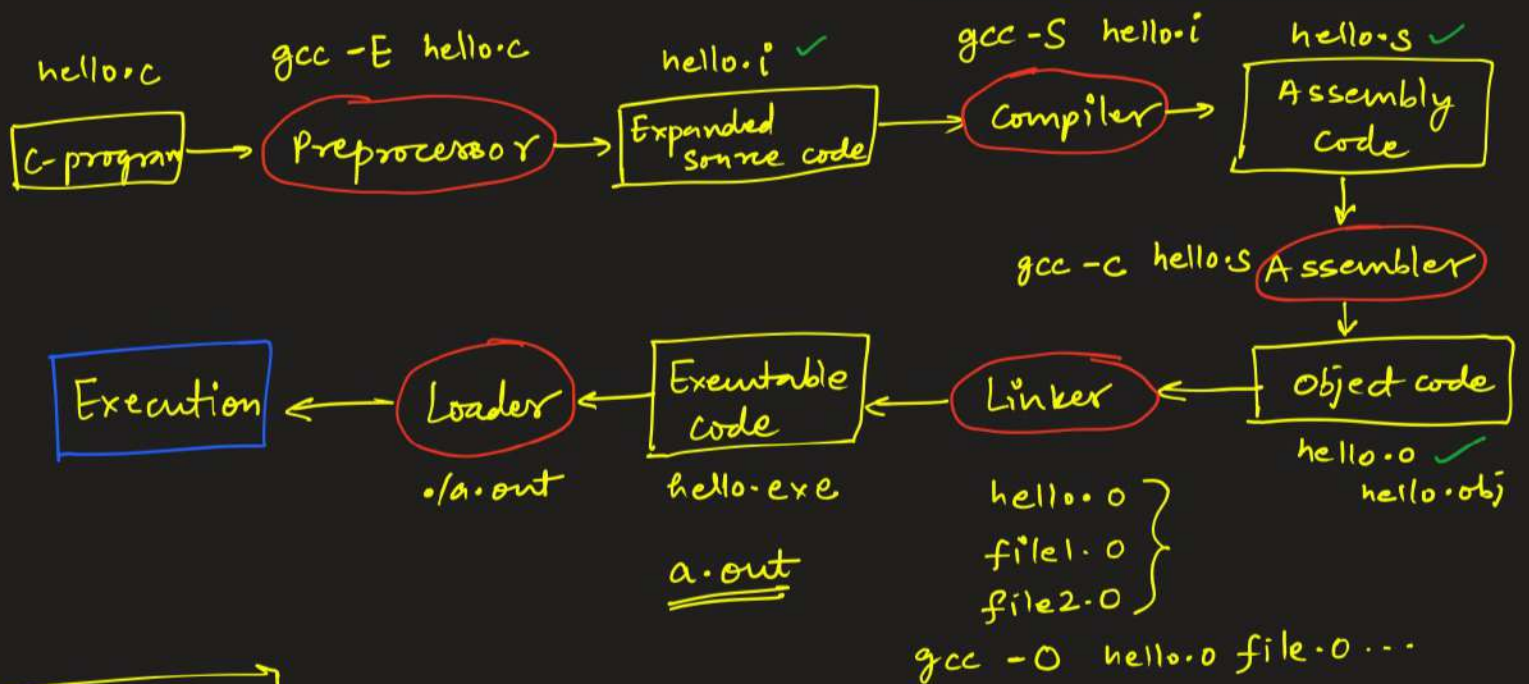
GCC
↑

The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.

The following are the phases through which our program passes before being transformed into an executable form:

- **Preprocessor** ✓
- **Compiler** ✓
- **Assembler** ✓
- **Linker** ✓





`gcc hello.c`
`-o a.out`

hello.s, hello.i, hello.o are all intermediate temporary files which are deleted automatically when we invoke gcc. If we want to keep them, we can use the following flags along with the command:

```
gcc -Wall -save-temps hello.c
```

