Compiling a C program:- Behind the Scenes

C is a high-level language and it needs a compiler to convert it into an executable code so that the program can be run on our machine.

## ****What goes inside the compilation process?****

Compiler converts a C program into an executable. There are four phases for a C program to become an executable:

1. Pre-processing
2. Compilation
3. Assembly
4. Linking

## ****Pre-processing****

This is the first phase through which source code is passed. This phase include:

* Removal of Comments
* Expansion of Macros
* Expansion of the included files.
* Conditional compilation

The preprocessed output is stored in the **filename.i**.

This file is in pure high level language

## ****Compiling****

The next step is to compile filename.i and produce an; intermediate compiled output file **filename.s**. This file is in assembly level instructions.

## ****Assembly****

In this phase the filename.s is taken as input and turned into **filename.o** by assembler. This file contain machine level instructions. At this phase, only existing code is converted into machine language, the function calls like printf() are not resolved.

## ****Linking****

This is the final phase in which all the linking of function calls with their definitions are done. Linker knows where all these functions are implemented. Linker does some extra work also, it adds some extra code to our program which is required when the program starts and ends.

For example, there is a code which is required for setting up the environment like passing command line arguments. This task can be easily verified by using **$size filename.o** and **$size filename**.

 Through these commands, we know that how output file increases from an object file to an executable file. This is because of the extra code that linker adds with our program.