The compilation process in C can be divided into several phases, each with specific tasks and objectives. These phases ensure that the source code is translated into an efficient and executable form while also checking for errors and allowing for modularity. The phases of the process are-

1. Preprocessing
2. Compilation
3. Assembling
4. Linking

## ****Compilation In C****

The source code is used as the input by the compiler, which produces object code as the output. The entire C language compilation process consists of four steps, as mentioned above, i.e., pre-processing, compilation, assembly, and linking. We will discuss what goes on in each step in detail ahead.

### ****Step 1: Preprocessor****

The file extension (.c) is used to identify source code files, which are text files with code written in them. This source code is first given to the preprocessor, who expands it from there. After the code is extended, it is handed to the compiler for further processing. This phase includes the following-

#### ****Comments Removal:****

The [comments](https://unstop.com/blog/comments-in-c) are the portion of code that the pre-processor removes during compilation because they are not very useful to the machine. They are used in C programs to provide a general understanding of a particular statement or section of code. When the pre-processing stage is over, the comments in the program below will be deleted.

#### ****Macros Expansion:****

A macro in C programming is a section of code that has a given name and can be used repeatedly throughout a program. The #define preprocessor directive links a name to a value or a section of code and is used to define macros.

Before the program is compiled, the preprocessor substitutes the macro name with the appropriate value or code when it is used in the program. Note that using macros can make code simpler, easier to read, and require less typing.

#### ****File Inclusion:****

When pre-processing a C program, file inclusion refers to adding a different file containing some pre-written code. The #include directive is utilized here.

* File inclusion during pre-processing allows the #include<filename> directive to be replaced with the whole contents of the filename. Thus, the filename's content is added to the source code, producing a new intermediate file.
* ***Example***- We can integrate the <stdio.h> header file by inserting the #include directive at the beginning of our code. Contained within this header file are standard input/output functions, among which include the highly utilized scanf() and printf().

**NOTE:**The source program/ code file, which originally had a (.c) extension, is given a (.i) extension during this phase.

### ****Step 2: Compilation****

The Compilation phase in C involves translating the preprocessed source code into assembly or machine code and generating object files. Here's a breakdown of the actions that occur during this phase:

#### ****Translation:****

* The compiler translates the preprocessed C source code into an intermediate representation, typically assembly language or an intermediate machine-independent code.
* The translation process involves analyzing the code for syntax [errors](https://unstop.com/blog/types-of-errors-in-c) and generating corresponding machine-level instructions.

#### ****Code Optimization:****

* The compiler may perform various optimizations to improve the efficiency of the generated code.
* Optimization techniques include constant folding, loop unrolling, and inlining functions to reduce execution time or memory usage.

#### ****Generation of Object Files:****

* The result of the compilation phase is one or more object files (with a .o extension on Unix-like systems or .obj on Windows).
* Object files contain machine code specific to the target architecture and may also include additional information, such as symbols and relocation entries.

### ****Step 3: Assembler****

The compiler sends the assembly code (i.e., a simple English language used to write low-level instructions) to the assembler, which then turns it into object code. In other words, the assembler helps in converting the assembly file into an object file containing machine-level code. In this step, the file's extension changes to (.obj).

### ****Step 4: Linking****

A linker is a tool used to connect every component of a program in the correct execution order. After the code has gone through this phase/ stage of compilation in C, it is in the executable machine code format. Note that if a program's code proves to be excessively lengthy, it may be necessary to break its contents into two distinct documents. These divided segments can then undergo separate construction processes before ultimately converging via an intermediary module known as the Linker.

The linker is a crucial component of the C language compilation process. Suppose your C program features a header file and integrates any function specified within this essential script. In that case, the linker shall proceed to bundle together both your object code and every pertinent library-based counterpart.