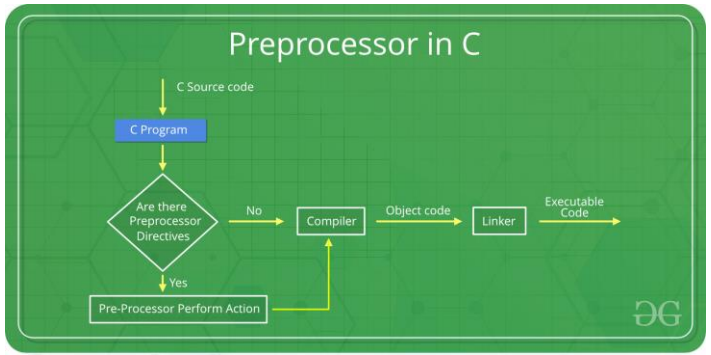
C Programming Language

How C Code Gets Compiled: Preprocessor to Executable



Microsoft

Preetam Sir
(Microsoft Cybersecurity Architect)

What is programming?

The term programming means to create (or develop) software, which is also called a program.

Computer programs, known as software, are instructions that tell a computer what to do.

Every Computer language is designed for some specific purpose. For example, Fortran was designed for scientific and mathematical calculations, COBOL (Common business Oriented Language) was designed for business applications. Similarly, C language was developed for programming in the operating system called UNIX.

Why do we use C?

C language is used to create applications or software.

Initially, C was developed to create an operating system called UNIX.

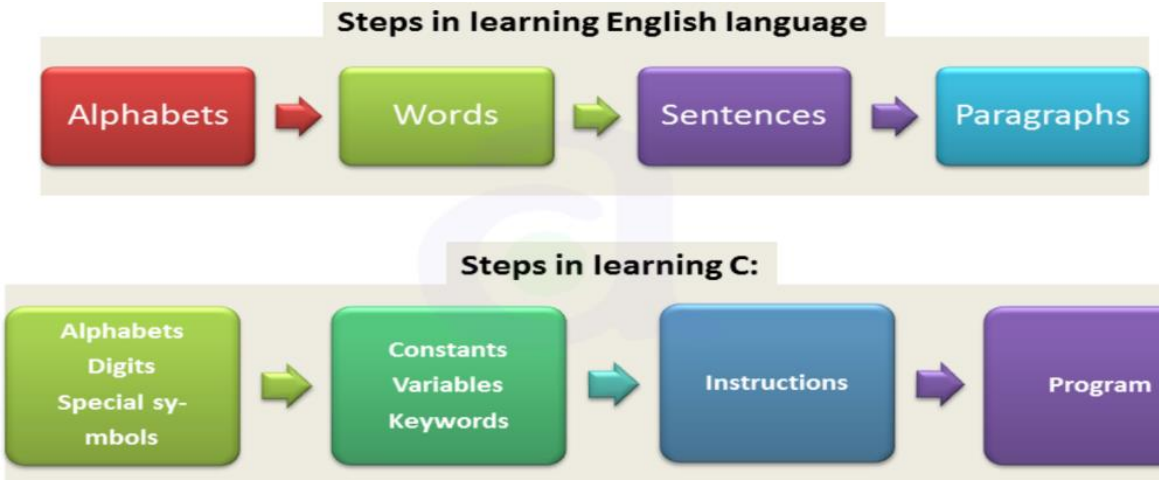
The popular software like **Linux OS**, **PHP** & **MySQL** are created using C language.

What is c?

C is a middle-level and general-purpose programming language that is ideal for developing firmware or portable applications.

Understanding C Language and English Language :

Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables, and keywords are constructed, and finally how are these combined to form an instruction? A group of instructions would be combined later on to form a program.

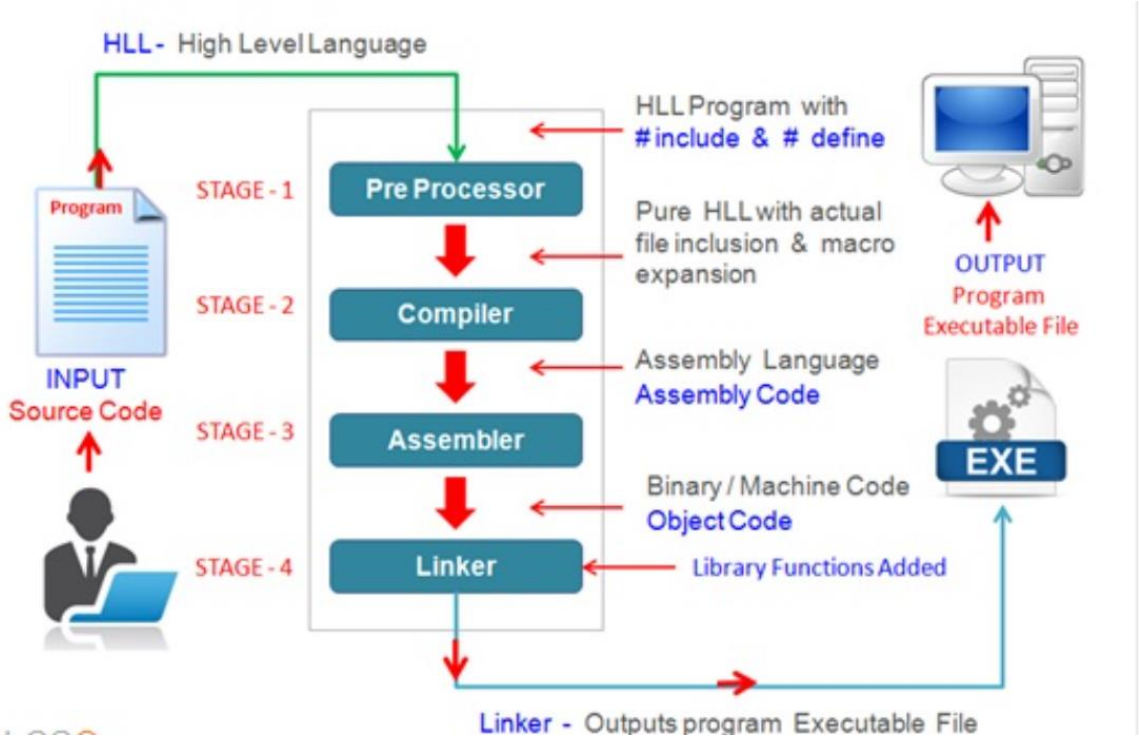


What is Program Compilation?

Compilation is the process of translating high-level source code into low-level machine code

- The **output** of compilation is binary code (machine-readable format).
- **Computers** can only execute instructions written in **binary** due to their hardware architecture.

Therefore, any program written in a high-level language must be **converted** into machine code. This machine code is what the **processor understands and executes**.



Program Compilation Stages

Compilation Stage	Accepts Input	Gives Output	Result
1. Pre-processing	Program Source Code	Pre-processed Code	Resolve #include, expand #define, remove comments
2. Compilation	Pre-processed Code	Assembly Code Instructions	Compiler produces Assembly Code
3. Assembly	Assembly Code Instructions	Object Code File	Assembler produces Object Code
4. Linking	Object Code Files	Single Machine Code File	Linker links and produces Executable

What is Pre-Processing Stage?

The first stage of the compilation process is called the **preprocessor stage**.

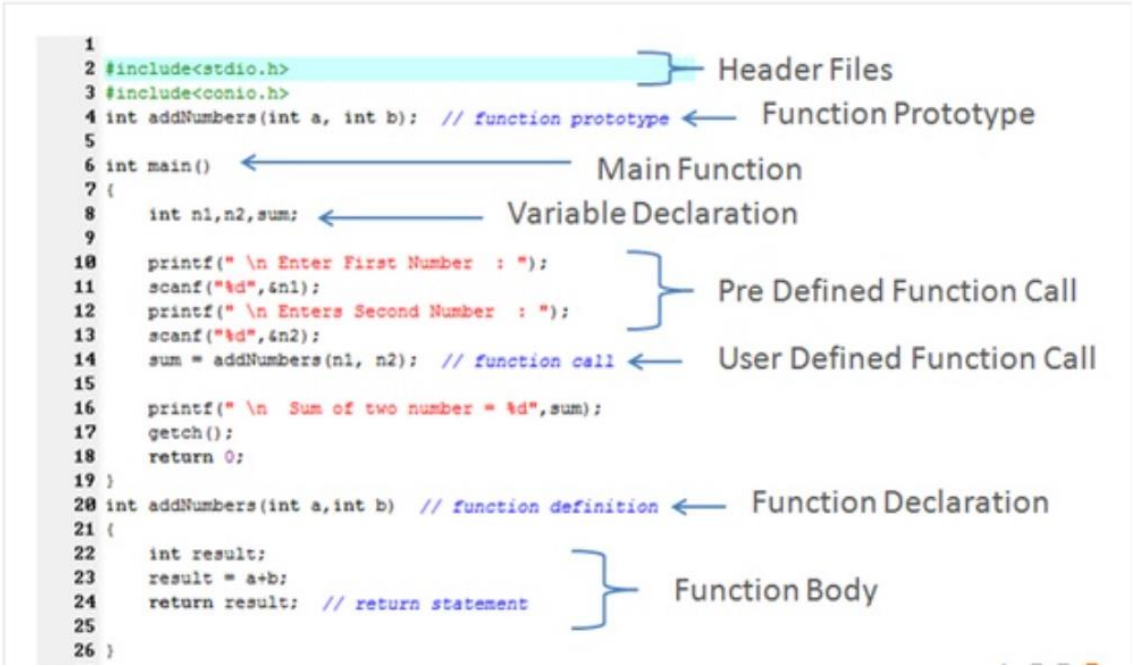
•It is also referred to as the **lexical analysis stage**.

Takes the program source code file as input.

The compiler scans the source code file for all **preprocessor directives**, such as:

•#include

•#define



What is Compilation stage

Compilation is the **second stage** of the program compilation process.

The compiler **accepts the preprocessed file as input**.

It **provides the assembly code as output** with a **.S** extension.

Converts high-level program instructions into equivalent **assembly code instructions**.

These assembly instructions are **platform dependent** and **compiled for a specific architecture**

Program Source Code

```
// www.learncomputerscienceonline.com
#include<stdio.h>
#include<conio.h>
int addNumbers(int a, int b);  // function prototype

int main()
{
    int n1,n2,sum;

    printf(" \n Enter First Number  : ");
    scanf("%d",&n1);
    printf(" \n Enters Second Number  : ");
    scanf("%d",&n2);
    sum = addNumbers(n1, n2);  // function call

    printf(" \n  Sum of two number = %d",sum);
    getch();
    return 0;
}
int addNumbers(int a,int b)  // function definition
{
    int result;
    result = a+b;
    return result;  // return statement

}
```

Program Assembly Code

```
DATA SEGMENT
     NUM1 DB 9H
     NUM2 DB 7H
     RESULT DB ?
ENDS

CODE SEGMENT
    ASSUME DS:DATA CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX

      MOV AL,NUM1
      ADD AL,NUM2

      MOV RESULT,AL

      MOV AH,4CH
      INT 21H
ENDS
END START
```

What is Assembly Stage

An **assembler** is used to convert this **assembly code** into **object code** (machine-level instructions).

The output of this stage is an **Object File**, usually with a .o or .obj extension.

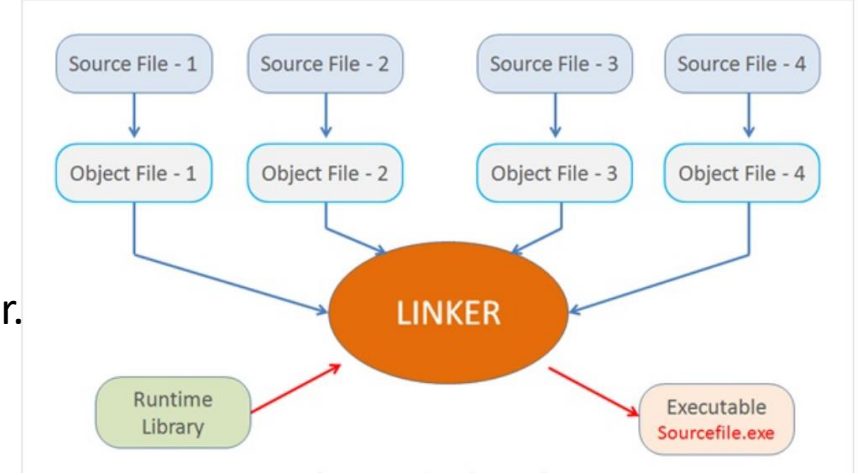
The **object file** contains **machine code**, but it is **not yet executable**.

```
        ST 1,[801]
        ST 0,[802]
TOP:    BEQ [802],10,BOT
        INCR [802]
        MUL [801],2,[803]
        ST [803],[801]
        JMP   TOP
BOT:    LD A,[801]
        CALL PRINT
```

```
00100101 11010011
00100100 11010100
10001010 01001001 11110000
01000100 01010100
01001000 10100111 10100011
11100101 10101011 00000010
00101001
11010101
11010100 10101000
10010001 01000100
```

Linking Stages

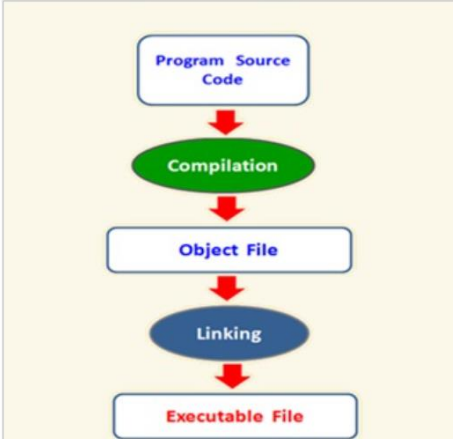
Linking is the **final (fourth) stage** of the compilation process.



The main function of linking is to:

- Produce a single **executable file**
- This is done by **linking all object code files** together.

If errors occur during this stage (like unresolved symbols), the linker throws **link-time errors**.



The **Linking Stage** is the **final step** in the program compilation process. It takes the object files and combines them into a complete, executable program.
It **resolves external references**, such as function calls or variables defined in other files.

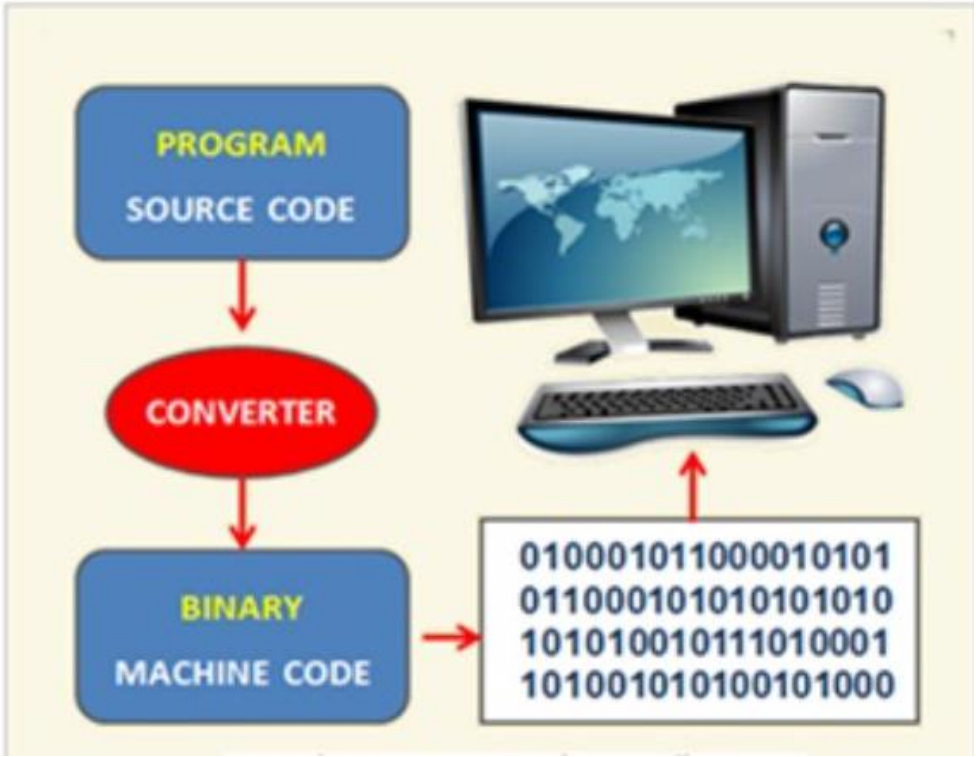
The linker **allocates memory** and sets up the final machine code layout.

How Computer Program are Compiled?

High Level Program

```
int main()
{
// Variable declaration
int a, b, sum;
// Take two numbers as input from the
user
scanf("%d %d", &a, &b);
// Add the numbers and assign the value
// to  some variable
sum = a + b;
// Use the calculated value
printf("%d\n", sum);
return 0;
// End of program
}
```

Low Level Machine Instructions



First C Program

```
#include <stdio.h>  // Preprocessor directive

int main()
 {       // Main function - program execution starts here
    printf("Hello, World!\n");  // Output function
    return 0;       // Exit the program successfully
}
```

C Token's

C Tokens: Character set, Identifiers, Keywords, constants, Data types, type qualifiers, Declaration and Initialization of variable

Token: The smallest individual units in a program are called tokens.

The 'C' tokens are classified as:

1. Character set
2. Keywords
3. Identifiers
4. Constants
5. Operators

Character set: Character set consists of_____

i) alphabet from A-Z or a-z

ii) digits from 0-9

iii) Special characters like (,), {,}, [,], , &, $, #, %, ^, !, ?, :, ;, ",’, .

iv) White space character: blank space

\b	backspace
\a	audible bell
\v	vertical tab
\t	horizontal tab
\f	form feed
\r	carriage return
\”	double quotes
\’	single quotes
\\	back slash

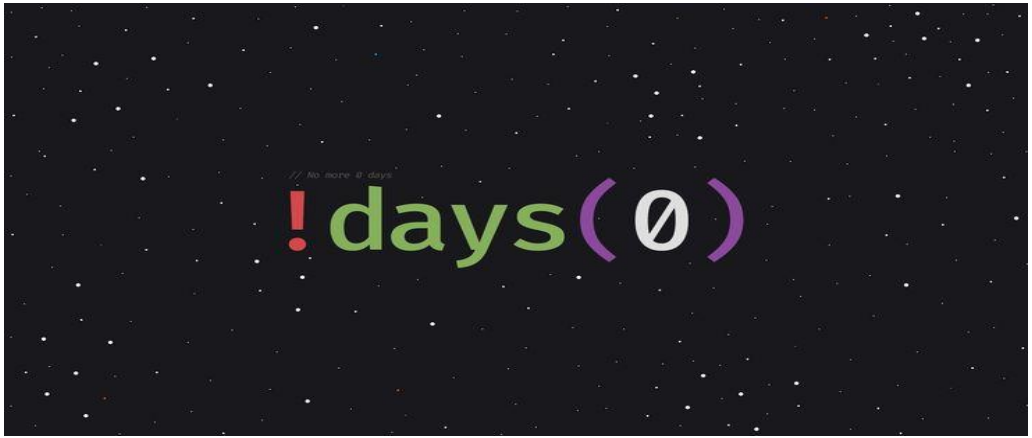
```c
#include <stdio.h>

int main() {
    int a = 5, b = 2;
    char ch = 'A';

    printf("Char: %c\n", ch);          // Letter
    printf("Sum: %d\n", a + b);        // Digits & Operators
    printf("Escape: \\ \" \n");        // Special characters & escape sequences
    return 0;
}
```

Output:

```
Char: A
Sum: 7
Escape: \ "
```

Thank You!

Hello World is just the beginning



-Preetam Sir