#Suppose you write any sample C program named as hello.c and executes it in Linux OS to get the a.out file. How many types of internal files are generated in system during the journey to hello.c to a.out. Try to display the assembly code and machine level code of your respective program Write the steps from starting I am completely newbie

Certainly! Let's walk through the steps of compiling a C program (hello.c) on a Linux system to generate the final executable (a.out), and how to view the assembly and machine-level code. We'll start with the basic C program and see how each step generates different types of files.

Ans:  
**Summary of Internal Files:**

* **Preprocessed file (hello.i)**: Contains expanded macros and included header files.
* **Assembly file (hello.s)**: Contains assembly language code.
* **Object file (hello.o)**: Contains machine code, but it's not executable on its own.
* **Executable file (a.out or specified name)**: The final runnable program.

1. Write the Sample C Program

Create a file named hello.c with the following simple C code:

c

Copy code

#include <stdio.h>

int main() {

    printf("Hello, World!\n");

    return 0;

}

2. Preprocessing

Command: gcc -E hello.c -o hello.i

Purpose: The preprocessor processes directives such as #include and #define. It generates a file with all macros expanded and header files included.

Output File: hello.i

View File:

cat hello.i

3. Compilation

Command: gcc -S hello.c -o hello.s

Purpose: The compiler translates the preprocessed code into assembly language.

Output File: hello.s

View File:

cat hello.s

4. Assembly

Command: gcc -c hello.c -o hello.o

Purpose: The assembler converts the assembly code into machine code and generates an object file.

Output File: hello.o

View Machine Code:

objdump -d hello.o

5. Linking

Command: gcc hello.c -o hello

Purpose: The linker combines the object file with any necessary libraries to produce the final executable.

Output File: hello (or a.out if not specified)

Complete Steps Summary

Write Source Code: hello.c

Preprocess:

gcc -E hello.c -o hello.i

Generates hello.i

Compile:

gcc -S hello.c -o hello.s

Generates hello.s

Assemble:

gcc -c hello.c -o hello.o

Generates hello.o

Link:

gcc hello.c -o hello

Generates hello (or a.out if no -o option)

Viewing Assembly and Machine Code

Assembly Code: View with cat hello.s

Machine Code: View with objdump -d hello.o

Example Commands

Preprocess:

gcc -E hello.c -o hello.i

cat hello.i

Compile:

gcc -S hello.c -o hello.s

cat hello.s

Assemble:

gcc -c hello.c -o hello.o

objdump -d hello.o

Link:

gcc hello.c -o hello

./hello

This process shows how your C program goes through several stages, each producing different types of files. Each stage represents a step in the compilation process, transforming the source code into a runnable executable.