Understanding What Happens When You Type gcc main.c

When you type `gcc main.c`, the `gcc` compiler (GNU Compiler Collection) goes through several steps to produce an executable file from your C source code. Let's explore these steps: **1. Preprocessing**
This step processes all the lines starting with `#`, such as `#include` and `#define`. The preprocessor expands macros and includes the content of header files into the source code.
Example: gcc -E main.c -o main.i **2. Compilation**
In this stage, the compiler translates the preprocessed code into assembly language instructions specific to the machine's architecture.
Example: gcc -S main.i -o main.s **3. Assembly**
The assembler converts the `.s` file (assembly code) into machine code and creates an object file (`.o`).
Example: gcc -c main.s -o main.o **4. Linking**
The linker combines the object file with necessary system libraries (like the standard C library) to create the final executable.
Example: gcc main.o -o a.out **Final Execution**
You can now run the program with: ./a.out **Diagram**
[C Source Code (main.c)] → Preprocessor → Compiler → Assembler → Linker → [Executable (a.out)] **Tip:** You can stop the compilation at any step using flags like `-E`, `-S`, or `-c`. **Conclusion:**
Understanding this pipeline helps you debug, optimize, and control the build process more effectively.