Introduction to the Compiler, Linker, and Libraries

Step 4: Compiling Your Source Code

The C++ compiler processes each `.cpp` source code file and performs two main tasks:

1. **Syntax Checking:** The compiler ensures that your C++ code follows the rules of the C++ language. If it does not, the compiler generates an error.
2. **Translation:** The compiler translates your C++ code into machine language instructions, which are stored in an intermediate file called an **object file**. These object files also contain additional data required for later steps.

Object files are named `name.o` (on Unix-based systems) or `name.obj` (on Windows), where `name` matches the original `.cpp` file.

Step 5: Linking Object Files and Libraries

After the compilation step, the **linker** takes over. The linker combines all the object files to produce the final output file, called the **executable**. This process is known as **linking**.

The linker performs the following tasks:

1. **Validates Object Files:** Ensures all the object files are valid and compatible.
2. **Resolves Cross-File Dependencies:** If a function or variable is defined in one `.cpp` file and used in another, the linker connects the reference to its definition. If it cannot do so, a linker error occurs, and the process halts.
3. **Links Library Files:** The linker incorporates precompiled code from libraries. A **library** is a collection of precompiled functions packaged for use in other programs.

Libraries in C++

C++ comes with the **C++ Standard Library**, which provides many useful capabilities for programs. Additionally, you can link other libraries, such as those provided by your operating system or third-party libraries (e.g., from GitHub).

The Build Process

The term **building** encompasses both the compiling and linking steps. For large or complex projects, **build automation tools** (e.g., `make` or `build2`) are often used to simplify the process. These tools can also automate running tests and other tasks.

Step 6: Run and Debug

Once the program is built, you can run it to see if it behaves as expected. If it does not, you must **debug** the program to identify and fix errors. Debugging is a critical part of the software development process and helps ensure your program operates correctly.