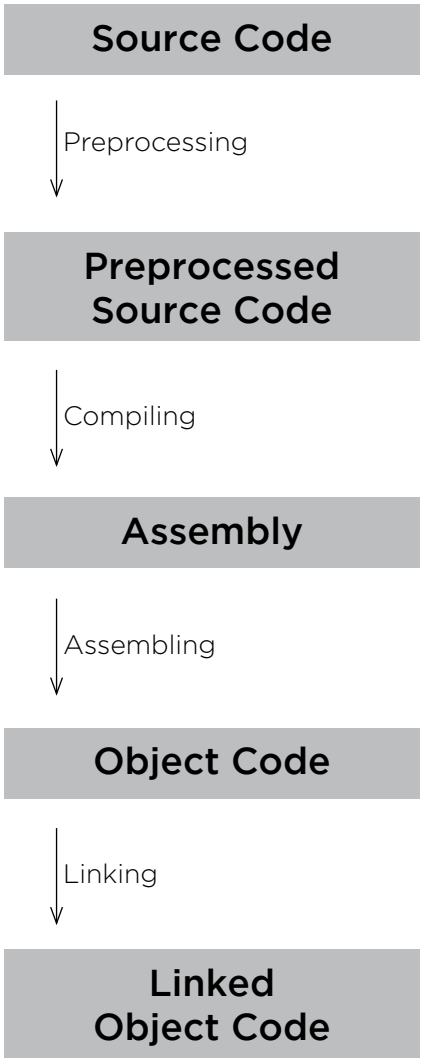
CS50 Compiling

Overview

Compiling is the process of translating source code, which is the code that you write in a programming language like C, and translating it into **object code** (also known as machine code), which are the 0s and 1s that actually tell a computer how to run a program. The command `make` itself is not a compiler; instead, `make` calls upon the underlying compiler, `clang`, in order to compile C source code into object code.

Key Terms

- compiling
- object code
- preprocessing
- assembly
- linking



Preprocessing

The entire compilation process can be broken down into four steps. The first step is **preprocessing**, performed by a program called the preprocessor. Any source code in C that begins with a `#` is a signal to the preprocessor to perform some action.

For example, `#include` tells the preprocessor to literally include the contents of a different file in the pre-processed file. When a program includes a line like `#include <stdio.h>` in the source code, the preprocessor generates a new file (still in C, and still considered source code), but with the `#include` line replaced by the entire contents of `stdio.h`.

Compiling

After the preprocessor produces preprocessed source code, the next step is to compile (using a program called a compiler) C code into a lower-level programming language known as **assembly**.

Assembly has far fewer different types of operations than C does, but by using them in conjunction, can still perform the same tasks that C can. By translating C code into assembly code, the compiler takes a program and brings it closer to a language that a computer can actually understand. The term "compiling" can refer to the entire process of translating source code to object code, but it can also be used to refer to this specific step of the compilation process.

Assembling

Once source code has been translated into assembly code, the next step is to turn the assembly code into object code: the sequence of 0s and 1s that a computer's central processing unit (CPU) can understand as instructions for how to execute the program. This translation is done with a program called the assembler. If there's only one file that needs to be compiled from source code to object code, the compilation process is over now. However, in other cases, an additional step is required: linking.

Linking

If a program has multiple files that need to be combined into a single object code file (such as if a program includes libraries like `math.h` or `cs50.h`), then one final step is required in the compilation process: linking. The linker takes multiple different object code files, and combines them into a single object code file that can be executed. For example, linking the CS50 Library during compilation is how the resulting object code knows how to execute functions like `GetInt()` or `GetString()`.

When linking files together, only one file can have a `main` function, so that the program knows where to begin.