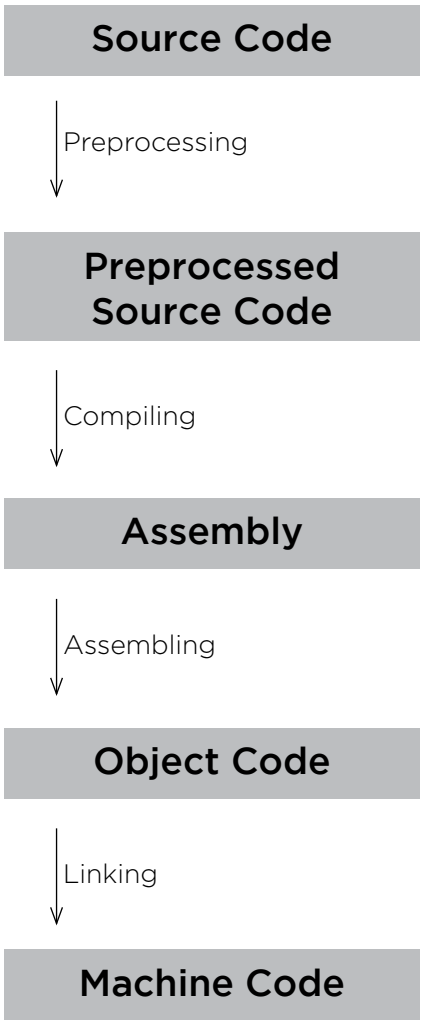
CS50 Compiling

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 **machine 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. Although the command `make` is used to compile code, `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
- machine code
- preprocessing
- assembly
- object code
- 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 preprocessed 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 much 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. This translation is done with a program called the assembler.

Object code is essentially machine code with some non-machine code symbols. If there's only one file that needs to be compiled from source code to machine code, the compilation process is over now. However, if there are multiple files to be compiled, a file's object code only represents part of the program and an additional step is required. The object code file's non-machine code symbols denote how the file fits with the other parts of the program. The entire program is put together in a process called linking.

Linking

If a program has multiple files that need to be combined into a single machine code file (such as if a program includes multiple files or 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 machine 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 `get_int()` or `get_string()`. It is important to note that only one file can have a `main` function so that the program knows where to begin.

© 2018