Compilation System: Recap

We call a C program, like *hello.c*, a high-level program because it can be read, written, and understood by human beings. However, to run *hello.c* on a computing system the individual C statements comprising that program must be translated into a sequence of low-level machine-language instructions. These instructions are then packaged in an executable object program, executable for short, and stored as a binary disk file.

The compilation system, compiler for short, is in charge of transforming a high-level C program into a executable object program. It does so in 4 stages:

1. Preprocessing. The preprocessor (*cpp*) modifies the original C program according to directives that begin with the '#' character. The result is another C program stored in a file typically with the *.i* suffix.
2. Compilation. The compiler (*cc1*) translates the text file *hello.i* into the text file *hello.s*, which contains an assembly-language program.
3. Assembly. The assembler (*as*) translate *hello.s* into machine-language instructions packaged in a relocatable object file, object file for short, *hello.o*.
4. Linking. The linker (*ld*) takes *hello.o* and merges with any library functions that *hello.c* might call to create an executable. On Unix systems such executables do not have a suffix; on other systems it may have a *.exe* suffix.