Linking

/run/media/lijunjie/资料/Learning Video/writing-running-fixing-code/02_compiling-and-running/03_compiler-details/05_linking_instructions.html

The final step of the process is to *link* the program. Linking the program takes one or more object files and combines them together with various *libraries* , as well as some startup code, and produces the actual executable binary. The object files refer to functions by name, and the linker's job is to resolve these references—finding the matching definition. If the linker cannot find the definition for a name (called a "symbol") that is used, or if the same symbol is defined multiple times, it will report an error.

Linker errors—indicated by the fact that they are reported by *ld* (the linker's name)—are typically less common than other compiler errors. If you encounter an unresolved symbol error, it means that you either did not define the symbol, did not include the object file that defines the symbol in the link, or that the symbol was specified as only visible inside that object file (which can be done by using the *static* keyword—a feature we will not delve into). If you encounter errors from duplicate definitions of a symbol, first make sure you did not try to name two different functions with the same name. Next, make sure you did not include any files twice on the compilation command line. Finally, make sure you do not *#include* a *.c* file—only header files—and that you only include function *prototypes* in the header file, not the function's definition (there are some advanced exceptions to these rules, but if you are at that stage, you should understand the linking process and static keyword well enough to fix any problems).

Sometimes you may wish to use an external library—a collection of functions that are already written and packaged up to use for some purpose (e.g., drawing graphics, playing sound, advanced math, or many other purposes). The C library is one such example, which is linked in by default. For other libraries, you must specifically request that the linker link with the library with the -l command line option, specifying the name of the library right after the l.

If all goes well, the linker will resolve the all the symbol references, and combined the various object files and libraries together into an executable binary.