- The linker's job is to pull together different pieces of a program. If it spots something it doesn't recognize, such as `retrun`, it assumes, "Hey, maybe it's something from another part of the program." So the error slides by. But, when the linker tries to look for the unrecognized word, it hoists its error flags high in the full breeze.

All about errors!

A common programming axiom is that you don't write computer programs as much as you remove errors from them. Errors are everywhere, and removing them is why it can take years to write good software.

Compiler errors: The most common error, initially discovered by the compiler as it tries to churn the text you write into instructions the computer can understand. These errors are the friendly ones, generally self-explanatory with line numbers and all the trimmings. The errors are caught before the program is built.

Linker errors: Primarily involve misspelled commands. In advanced C programming, when you're working with several source files, or modules, to create a larger program, linker errors may involve missing modules. Also, if your linker requires some "library" file and it can't be found, another type of error message is displayed. Pieces of the program are built, but errors prevent it from them being glued together.

Run-time errors: Generated by the program when it runs. They aren't bugs; instead, they're things that look totally acceptable to the compiler and linker but just don't do quite what you intended. (This happens often in C.) The most common run-time error is a *null pointer assignment.* You aggravate over this one later. The program is built, but usually gets shut down by the operating system when it's run.

Bugs: The final type of error you encounter. The compiler diligently creates the program you wrote, but whether that program does what you intended is up to the test. If it doesn't, you must work on the source code some more. Bugs include everything from things that work slowly to ones that work unintentionally or not at all. These are the hardest things to figure out and are usually your highest source of frustration. The program is built and runs, but it doesn't behave the way you think it would.