file in the program, make can determine which files are out of date. It then invokes the commands necessary to rebuild the program.

If you want to give make a try, here are a few details you'll need to know:

- Each command in a makefile must be preceded by a tab character, not a series of spaces. (In our example, the commands appear to be indented eight spaces, but it's actually a single tab character.)
- A makefile is normally stored in a file named `Makefile` (or `makefile`). When the make utility is used, it automatically checks the current directory for a file with one of these names.
- To invoke make, use the command

  ```
  make target
  ```

  where *target* is one of the targets listed in the makefile. To build the `justify` executable using our makefile, we would use the command

  ```
  make justify
  ```

- If no target is specified when make is invoked, it will build the target of the first rule. For example, the command

  ```
  make
  ```

  will build the `justify` executable, since `justify` is the first target in our makefile. Except for this special property of the first rule, the order of rules in a makefile is arbitrary.

make is complicated enough that entire books have been written about it, so we won't attempt to delve further into its intricacies. Let's just say that real makefiles aren't usually as easy to understand as our example. There are numerous techniques that reduce the amount of redundancy in makefiles and make them easier to modify; at the same time, though, these techniques greatly reduce their readability.

Not everyone uses makefiles, by the way. Other program maintenance tools are also popular, including the "project files" supported by some integrated development environments.

Errors During Linking

Some errors that can't be detected during compilation will be found during linking. In particular, if the definition of a function or variable is missing from a program, the linker will be unable to resolve external references to it, causing a message such as "*undefined symbol*" or "*undefined reference.*"

Errors detected by the linker are usually easy to fix. Here are some of the most common causes:

- ***Misspellings.*** If the name of a variable or function is misspelled, the linker will report it as missing. For example, if the function `read_char` is defined