character. Regardless of whether an input file contains a carriage-return character, a line-feed character, or both, a library function such as `getc` will return a single new-line character. The output functions perform the reverse translation. If a program calls a library function to write a new-line character to a file, the function will translate the character into the appropriate end-of-line marker. C's approach makes programs more portable and easier to write; we can work with text files without having to worry about how end-of-line is actually represented. Note that input/output performed on a file opened in binary mode isn't subject to any character translation—carriage return and line feed are treated the same as the other characters.

Q: I'm writing a program that needs to save data in a file, to be read later by another program. Is it better to store the data in text form or binary form? [p. 542]

A: That depends. If the data is all text to start with, there's not much difference. If the data contains numbers, however, the decision is tougher.

Binary form is usually preferable, since it can be read and written quickly. Numbers are already in binary form when stored in memory, so copying them to a file is easy. Writing numbers in text form is much slower, since each number must be converted (usually by `fprintf`) to character form. Reading the file later will also take more time, since numbers will have to be converted from text form back to binary. Moreover, storing data in binary form often saves space, as we saw in Section 22.1.

Binary files have two disadvantages, however. They're hard for humans to read, which can hamper debugging. Also, binary files generally aren't portable from one system to another, since different kinds of computers store data in different ways. For instance, some machines store `int` values using two bytes but others use four bytes. There's also the issue of byte order (big-endian versus little-endian).

Q: C programs for UNIX never seem to use the letter `b` in the mode string, even when the file being opened is binary. What gives? [p. 544]

A: In UNIX, text files and binary files have exactly the same format, so there's never any need to use `b`. UNIX programmers should still include the `b`, however, so that their programs will be more portable to other operating systems.

Q: I've seen programs that call `fopen` and put the letter `t` in the mode string. What does `t` mean?

A: The C standard allows additional characters to appear in the mode string, provided that they follow `r`, `w`, `a`, `b`, or `+`. Some compilers allow the use of `t` to indicate that a file is to be opened in text mode instead of binary mode. Of course, text mode is the default anyway, so `t` adds nothing. Whenever possible, it's best to avoid using `t` and other nonportable features.

Q: Why bother to call `fclose` to close a file? Isn't it true that all open files are closed automatically when a program terminates? [p. 545]