undefined.) There are two ways to avoid the problem. One is to use \\ instead of \:

```
fopen("c:\\project\\test1.dat", "r")
```

The other technique is even easier—just use the / character instead of \:

```
fopen("c:/project/test1.dat", "r")
```

Windows will happily accept / instead of \ as the directory separator.

`fopen` returns a file pointer that the program can (and usually will) save in a variable and use later whenever it needs to perform an operation on the file. Here's a typical call of `fopen`, where `fp` is a variable of type `FILE *`:

```
fp = fopen("in.dat", "r");   /* opens in.dat for reading */
```

When the program calls an input function to read from `in.dat` later, it will supply `fp` as an argument.

When it can't open a file, `fopen` returns a null pointer. Perhaps the file doesn't exist, or it's in the wrong place, or we don't have permission to open it.

Never assume that a file can be opened; always test the return value of `fopen` to make sure it's not a null pointer.

Modes

Which mode string we'll pass to `fopen` depends not only on what operations we plan to perform on the file later but also on whether the file contains text or binary data. To open a text file, we'd use one of the mode strings in Table 22.2.

Table 22.2 Mode Strings for Text Files

String	*Meaning*
`"r"`	Open for reading
`"w"`	Open for writing (file need not exist)
`"a"`	Open for appending (file need not exist)
`"r+"`	Open for reading and writing, starting at beginning
`"w+"`	Open for reading and writing (truncate if file exists)
`"a+"`	Open for reading and writing (append if file exists)

Q&A

When we use `fopen` to open a binary file, we'll need to include the letter `b` in the mode string. Table 22.3 lists mode strings for binary files.

From Tables 22.2 and 22.3, we see that `<stdio.h>` distinguishes between *writing* data and *appending* data. When data is written to a file, it normally overwrites what was previously there. When a file is opened for appending, however, data written to the file is added at the end, thus preserving the file's original contents.

By the way, special rules apply when a file is opened for both reading and writing (the mode string contains the + character). We can't switch from reading to writ-