Table 22.3
Mode Strings for Binary Files

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

file-positioning functions ➤22.7

ing without first calling a file-positioning function unless the reading operation encountered the end of the file. Also, we can't switch from writing to reading without either calling `fflush` (covered later in this section) or calling a file-positioning function.

Closing a File

```
int fclose(FILE *stream);
```

fclose

The `fclose` function allows a program to close a file that it's no longer using. The argument to `fclose` must be a file pointer obtained from a call of `fopen` or `freopen` (discussed later in this section). `fclose` returns zero if the file was closed successfully; otherwise, it returns the error code `EOF` (a macro defined in `<stdio.h>`).

Q&A

To show how `fopen` and `fclose` are used in practice, here's the outline of a program that opens the file `example.dat` for reading, checks that it was opened successfully, then closes it before terminating:

```
#include <stdio.h>
#include <stdlib.h>

#define FILE_NAME "example.dat"

int main(void)
{
  FILE *fp;

  fp = fopen(FILE_NAME, "r");
  if (fp == NULL) {
    printf("Can't open %s\n", FILE_NAME);
    exit(EXIT_FAILURE);
  }
  ...
  fclose(fp);
  return 0;
}
```

Of course, C programmers being the way they are, it's not unusual to see the call of `fopen` combined with the declaration of `fp`:

```
FILE *fp = fopen(FILE_NAME, "r");
```