**Exercise 8-1.** Rewrite the program cat from [Chapter 7](file:///C:\Users\Valentin\AppData\Local\Temp\Rar$EXa1580.22235\tcpl\chapter7.html) using read, write, open, and close instead of their standard library equivalents. Perform experiments to determine the relative speeds of the two versions.

**Exercise 8-2.** Rewrite fopen and \_fillbuf with fields instead of explicit bit operations. Compare code size and execution speed.

**Exercise 8-3.** Design and write \_flushbuf, fflush, and fclose.

**Exercise 8-4.** The standard library function

int fseek(FILE \*fp, long offset, int origin)

is identical to lseek except that fp is a file pointer instead of a file descriptor and return value is an int status, not a position. Write fseek. Make sure that your fseek coordinates properly with the buffering done for the other functions of the library.

**Exercise 8-5.** Modify the fsize program to print the other information contained in the inode entry.

**Exercise 8-6.** The standard library function calloc(n,size) returns a pointer to n objects of size size, with the storage initialized to zero. Write calloc, by calling malloc or by modifying it.

**Exercise 8-7.** malloc accepts a size request without checking its plausibility; free believes that the block it is asked to free contains a valid size field. Improve these routines so they make more pains with error checking.

**Exercise 8-8.** Write a routine bfree(p,n) that will free any arbitrary block p of n characters into the free list maintained by malloc and free. By using bfree, a user can add a static or external array to the free list at any time.