When using `setvbuf` or `setbuf`, be sure to close the stream before its buffer is deallocated. In particular, if the buffer is local to a function and has automatic storage duration, be sure to close the stream before the function returns.

Miscellaneous File Operations

```
int remove(const char *filename);
int rename(const char *old, const char *new);
```

The functions `remove` and `rename` allow a program to perform basic file management operations. Unlike most other functions in this section, `remove` and `rename` work with file *names* instead of file *pointers*. Both functions return zero if they succeed and a nonzero value if they fail.

remove

`remove` deletes a file:

```
remove("foo");              /* deletes the file named "foo" */
```

If a program uses `fopen` (instead of `tmpfile`) to create a temporary file, it can use `remove` to delete the file before the program terminates. Be sure that the file to be removed has been closed; the effect of removing a file that's currently open is implementation-defined.

rename

`rename` changes the name of a file:

```
rename("foo", "bar");    /* renames "foo" to "bar" */
```

`rename` is handy for renaming a temporary file created using `fopen` if a program should decide to make it permanent. If a file with the new name already exists, the effect is implementation-defined.

If the file to be renamed is open, be sure to close it before calling `rename`; the function may fail if asked to rename an open file.

22.3 Formatted I/O

In this section, we'll examine library functions that use format strings to control reading and writing. These functions, which include our old friends `printf` and `scanf`, have the ability to convert data from character form to numeric form during input and from numeric form to character form during output. None of the other I/O functions can do such conversions.