or the test against NULL:

```
if ((fp = fopen(FILE_NAME, "r")) == NULL) ...
```

Attaching a File to an Open Stream

```
FILE *freopen(const char * restrict filename,
              const char * restrict mode,
              FILE * restrict stream);
```

freopen

`freopen` attaches a different file to a stream that's already open. The most common use of `freopen` is to associate a file with one of the standard streams (`stdin`, `stdout`, or `stderr`). To cause a program to begin writing to the file `foo`, for instance, we could use the following call of `freopen`:

```
if (freopen("foo", "w", stdout) == NULL) {
  /* error; foo can't be opened */
}
```

After closing any file previously associated with `stdout` (by command-line redirection or a previous call of `freopen`), `freopen` will open `foo` and associate it with `stdout`.

`freopen`'s normal return value is its third argument (a file pointer). If it can't open the new file, `freopen` returns a null pointer. (`freopen` ignores the error if the old file can't be closed.)

C99

C99 adds a new twist. If `filename` is a null pointer, `freopen` attempts to change the stream's mode to that specified by the `mode` parameter. Implementations aren't required to support this feature, however: if they do, they may place restrictions on which mode changes are permitted.

Obtaining File Names from the Command Line

When we're writing a program that will need to open a file, one problem soon becomes apparent: how do we supply the file name to the program? Building file names into the program itself doesn't provide much flexibility, and prompting the

Q&A

user to enter file names can be awkward. Often, the best solution is to have the program obtain file names from the command line. When we execute a program named `demo`, for example, we might supply it with file names by putting them on the command line:

```
demo names.dat dates.dat
```

In Section 13.7, we saw how to access command-line arguments by defining `main` as a function with two parameters:

```
int main(int argc, char *argv[])
{
  ...
}
```