file **1**

`file.` A *file* is a stream of information that TEX interprets or creates. Files are managed by the operating system that supervises your TEX run. TEX deals with files in four different contexts:

1) A "source file" is one that TEX reads with its "eyes" (see "anatomy of TEX", p. '`\anatomy`') and interprets according to its ordinary rules. Your primary input file—the one you specify after '`**`' or on the command line when you invoke TEX—is a source file, and so is any file that you call for with an `\input` command (p. '`\input`').
2) A "result file" is one that contains the results of running TEX. A TEX run creates two result files: the `.dvi` file and the log file. The `.dvi` file contains the information needed to print your document; the log file contains a record of what happened during the run, including any error messages that TEX generated. If your primary source file is named `screed.tex`, your `.dvi` file and log file will be named `screed.dvi` and `screed.log`.[1]
3) To read from a file with the `\read` command (p. '`\read`') you need to associate the file with an input stream. You can have up to 16 input streams active at once, numbered 0–15. The `\read` command reads a single line and makes it the value of a designated control sequence, so reading with `\read` is very different from reading with `\input` (which brings in an entire file). TEX takes any input stream number not between 0 and 15 to refer to the terminal, so '`\read16`', say, reads the next line that you type at the terminal.
4) To write to a file with the `\write` command (p. '`\write`') you need to associate the file with an output stream. You can have up to 16 output streams active at once, numbered 0–15. Input and output streams are independent. Anything sent to an output stream with a negative number goes to the log file; anything sent to an output stream with a number greater than 15 goes both to the log file and to the terminal. Thus '`\write16`', say, writes a line on the terminal and also sends that line to the log file.

You must open a stream file before you can use it. An input stream file is opened with an `\openin` command (p. '`\openin`') and an output stream file is opened with an `\openout` command (p. '`\openout`'). For tidiness you should close a stream file when you're done with it, although TEX will do that at the end of the run if you don't. The two commands for closing a stream file are `\closein` (p. '`\closein`') and `\closeout` (p. '`\closeout`'). An advantage of closing a stream when you're done with it is that you can then reuse the stream for a different file. Doing this can be essential when you're reading a long sequence of files.

[1] This is the usual convention, but particular implementations of TEX are free to change it.

Although you can assign numbers yourself to input and output streams, it's better to do it with the `\newread` and `\newwrite` (p. '`\@newwrite`') commands. You can have more than one stream associated with a particular file, but you'll get (probably undiagnosed) garbage unless all of the streams are input streams. Associating more than one stream with an input file can be useful when you want to use the same input file for two different purposes.

TEX ordinarily defers the actions of opening, writing to, or closing an output stream until it ships out a page with `\shipout` (see page 227 of *The TEXbook* for the details). This behavior applies even to messages written to the terminal with `\write`. But you can get TEX to perform an action on an output stream immediately by preceding the action command with `\immediate` (p. '`\immediate`'). For example:

```
\immediate\write16{Do not pass GO! Do not collect $200!}
```