wide-character function. I won't discuss the wide-character function further unless there's a significant difference between it and its "non-wide" counterpart.

Stream Orientation

Before we look at the input/output functions provided by `<wchar.h>`, it's important to understand ***stream orientation,*** a concept that doesn't exist in C89.

Every stream is either ***byte-oriented*** (the traditional orientation) or ***wide-oriented*** (data is written to the stream as wide characters). When a stream is first opened, it has no orientation. (In particular, the standard streams `stdin`, `stdout`, and `stderr` have no orientation at the beginning of program execution.) Performing an operation on the stream using a byte input/output function causes the stream to become byte-oriented; performing an operation using a wide-character input/output function causes the stream to become wide-oriented. The orientation of a stream can also be selected by calling the `fwide` function (described later in this section). A stream retains its orientation as long as it remains open. Calling the `freopen` function to reopen the stream will remove its orientation.

standard streams ➤*22.1*

`freopen` function ➤*22.2*

When wide characters are written to a wide-oriented stream, they are converted to multibyte characters before being stored in the file that is associated with the stream. Conversely, when input is read from a wide-oriented stream, the multibyte characters found in the stream are converted to wide characters. The multibyte encoding used in a file is similar to that used for characters and strings within a program, except that encodings used in files may contain embedded null bytes.

Each wide-oriented stream has an associated `mbstate_t` object, which keeps track of the stream's conversion state. An encoding error occurs when a wide character written to a stream doesn't correspond to any multibyte character, or when a sequence of characters read from a stream doesn't form a valid multibyte character. In either case, the value of the `EILSEQ` macro (defined in the `<errno.h>` header) is stored in the `errno` variable to indicate the nature of the error.

`errno` variable ➤*24.2*

Once a stream is byte-oriented, it's illegal to apply a wide-character input/output function to that stream. Similarly, it's illegal to apply a byte input/output function to a wide-oriented stream. Other stream functions may be applied to streams of either orientation, although there are a few special considerations for wide-oriented streams:

- Binary wide-oriented streams are subject to the file-positioning restrictions of both text and binary streams.
- After a file-positioning operation on a wide-oriented stream, a wide-character output function may end up overwriting part of a multibyte character. Doing so leaves the rest of the file in an indeterminate state.
- Calling `fgetpos` for a wide-oriented stream retrieves the stream's `mbstate_t` object as part of the `fpos_t` object associated with the stream. A later call of `fsetpos` using this `fpos_t` object will restore the `mbstate_t` object to its previous value.

`fgetpos` function ➤*22.7*

`fsetpos` function ➤*22.7*