std::FILE

Defined in header `<cstdio>`

```
typedef /* unspecified */ FILE;
```

Each `std::FILE` object denotes a C stream.

C standard (referenced by C++ standard) does not specify whether `std::FILE` is a complete object type. While it may be possible to copy a valid `std::FILE`, using a pointer to such a copy as an argument for an I/O function invokes unspecified behavior. In other words, `std::FILE` may be semantically non-copyable.

I/O streams can be used for both unformatted and formatted input and output. Furthermore, the functions that handle input and output can also be locale-sensitive, such that wide/multibyte conversions are performed as necessary.

Stream state

Besides the system-specific information necessary to access the device (*e.g.,* a POSIX file descriptor), each `std::FILE` object directly or indirectly holds the following:

1. Character width: unset, narrow, or wide.
2. Parse state for conversions between multibyte and wide characters (an object of type `std::mbstate_t`)
3. Buffering state: unbuffered, line-buffered, fully buffered.
4. The buffer, which may be replaced by an external, user-provided buffer.
5. I/O mode: input, output, or update (both input and output).
6. Binary/text mode indicator.
7. End-of-file status indicator.
8. Error status indicator.
9. File position indicator, accessible as an object of type `std::fpos_t`, which, for wide streams, includes parse state.
10. (C++17) Reentrant lock used to prevent data races when multiple threads read, write, position, or query the position of a stream.

Narrow and wide orientation

A newly opened stream has no orientation. The first call to `std::fwide` or to any I/O function establishes the orientation: a wide I/O function makes the stream wide-oriented; a narrow I/O function makes the stream narrow-oriented. Once set, the orientation can be changed with only `std::freopen`. Narrow I/O functions cannot be called on a wide-oriented stream; wide I/O functions cannot be called on a narrow-oriented stream. Wide I/O functions convert between wide and multibyte characters as if by calling `std::mbrtowc` or `std::wcrtomb` with the conversion state as described by the stream. Unlike the multibyte character strings that are valid in a program, multibyte character sequences in the file may contain embedded nulls and do not have to begin or end in the initial shift state.

The conversion state of a stream with wide orientation is established by the C locale that is installed at the time the stream's orientation is set.

Binary and text modes

A *text stream* is an ordered sequence of characters that can be composed into lines; a line can be decomposed into zero or more characters plus a terminating `'\n'` (“newline”) character. Whether the last line requires a terminating `'\n'` is implementation-defined. Furthermore, characters may have to be added, altered, or deleted on input and output to conform to the conventions for representing text in the OS (in particular, C streams on Windows OS convert `'\n'` to `'\r\n'` on output, and convert `'\r\n'` to `'\n'` on input).

Data read in from a text stream is guaranteed to compare equal to the data that were earlier written out to that stream only if each of the following is true:

- The data consist of only printing characters and/or the control characters `'\t'` and `'\n'` (in particular, on Windows OS, the character `'\0x1A'` terminates input).
- No `'\n'` character is immediately preceded by space characters (such space characters may disappear when such output is later read as input).
- The last character is `'\n'`.

A *binary stream* is an ordered sequence of characters that can transparently record internal data. Data read in from a binary stream always equal the data that were earlier written out to that stream, except that an implementation is allowed to append an indeterminate number of null characters to the end of the stream. A wide binary stream doesn't need to end in the initial shift state.

Notes

POSIX explicitly requires that the LC_CTYPE facet of the currently installed C locale be stored within the FILE object the moment the stream's orientation becomes wide; POSIX requires that this LC_CTYPE facet be used for all future I/O on this stream until the orientation is changed, regardless of any subsequent call to `std::setlocale`.

It is intended that each line of text be composed of data that are essentially human-readable. POSIX implementations do not distinguish between text and binary streams (there is no special mapping for `'\n'` or any other characters).

See also

basic_streambuf	abstracts a raw device (class template)
basic_filebuf	implements raw file device (class template)
stdin **stdout** **stderr**	expression of type FILE* associated with the input stream expression of type FILE* associated with the output stream expression of type FILE* associated with the error output stream (macro constant)

C documentation for **FILE**

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/io/c/FILE&oldid=149382"