



Linux Ubuntu 26.04 Manual Pages on command 'stdin.3'

\$ man stdin.3

stdin(3) Library Functions Manual stdin(3)

NAME

stdin, stdout, stderr - standard I/O streams

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <stdio.h>

extern FILE *stdin;

extern FILE *stdout;

extern FILE *stderr;
```

DESCRIPTION

Under normal circumstances every UNIX program has three streams opened for it when it starts up, one for input, one for output, and one for printing diagnostic or error messages. These are typically attached to the user's terminal (see [tty\(4\)](#)) but might instead refer to files or other devices, depending on what the parent process chose to set up. (See also the "Redirections" section of [sh\(1\)](#).)

The input stream is referred to as "standard input"; the output stream is referred to as "standard output"; and the error stream is referred to as "standard error". These terms are abbreviated to form the symbols used to refer to these files, namely `stdin`, `stdout`, and `stderr`.

Each of these symbols is a `stdio(3)` macro of type pointer to `FILE`, and can be used with functions like `fprintf(3)` or `fread(3)`.

Since `FILE`s are a buffering wrapper around UNIX file descriptors, the same underlying files

may also be accessed using the raw UNIX file interface, that is, the functions like `read(2)` and `lseek(2)`.

On program startup, the integer file descriptors associated with the streams `stdin`, `stdout`, and `stderr` are 0, 1, and 2, respectively. The preprocessor symbols `STDIN_FILENO`, `STDOUT_FILENO`, and `STDERR_FILENO` are defined with these values in `<unistd.h>`. (Applying `fopen(3)` to one of these streams can change the file descriptor number associated with the stream.)

Note that mixing use of `FILE`s and raw file descriptors can produce unexpected results and should generally be avoided. (For the masochistic among you: POSIX.1, section 8.2.3, describes in detail how this interaction is supposed to work.) A general rule is that file descriptors are handled in the kernel, while `stdio` is just a library. This means for example, that after an `exec(3)`, the child inherits all open file descriptors, but all old streams have become inaccessible.

Since the symbols `stdin`, `stdout`, and `stderr` are specified to be macros, assigning to them is nonportable. The standard streams can be made to refer to different files with help of the library function `freopen(3)`, specially introduced to make it possible to reassign `stdin`, `stdout`, and `stderr`. The standard streams are closed by a call to `exit(3)` and by normal program termination.

STANDARDS

C11, POSIX.1-2008.

The standards also stipulate that these three streams shall be open at program startup.

HISTORY

C89, POSIX.1-2001.

NOTES

The stream `stderr` is unbuffered. The stream `stdout` is line-buffered when it points to a terminal. Partial lines will not appear until `fflush(3)` or `exit(3)` is called, or a newline is printed. This can produce unexpected results, especially with debugging output. The buffering mode of the standard streams (or any other stream) can be changed using the `setbuf(3)` or `setvbuf(3)` call. Note that in case `stdin` is associated with a terminal, there may also be input buffering in the terminal driver, entirely unrelated to `stdio` buffering. (Indeed, normally terminal input is line buffered in the kernel.) This kernel input handling can be modified using calls like `tcsetattr(3)`; see also `stty(1)`, and `termios(3)`.

SEE ALSO

cs(1), sh(1), open(2), fopen(3), stdio(3)

Linux man-pages 6.17

2025-09-21

stdin(3)