NAME

write − write to a file descriptor

SYNOPSIS

#include <unistd.h>

ssize_t write(int *fd***, const void ****buf***, size_t** *count***);**

DESCRIPTION

write() writes up to *count* bytes from the buffer pointed *buf* to the file referred to by the file descriptor *fd*.

The number of bytes written may be less than *count* if, for example, there is insufficient space on the underlying physical medium, or the **RLIMIT_FSIZE** resource limit is encountered (see **setrlimit**(2)), or the call was interrupted by a signal handler after having written less than *count* bytes. (See also **pipe**(7).)

For a seekable file (i.e., one to which **lseek**(2) may be applied, for example, a regular file) writing takes place at the current file offset, and the file offset is incremented by the number of bytes actually written. If the file was **open**(2)ed with **O_APPEND**, the file offset is first set to the end of the file before writing. The adjustment of the file offset and the write operation are performed as an atomic step.

POSIX requires that a **read**(2) which can be proved to occur after a **write**() has returned returns the new data. Note that not all file systems are POSIX conforming.

RETURN VALUE

On success, the number of bytes written is returned (zero indicates nothing was written). On error, −1 is returned, and *errno* is set appropriately.

If *count* is zero and *fd* refers to a regular file, then **write**() may return a failure status if one of the errors below is detected. If no errors are detected, 0 will be returned without causing any other effect. If *count* is zero and *fd* refers to a file other than a regular file, the results are not specified.

ERRORS

EAGAIN
The file descriptor *fd* refers to a file other than a socket and has been marked non-blocking (**O_NONBLOCK**), and the write would block.

EAGAIN or **EWOULDBLOCK**
The file descriptor *fd* refers to a socket and has been marked non-blocking (**O_NONBLOCK**), and the write would block. POSIX.1-2001 allows either error to be returned for this case, and does not require these constants to have the same value, so a portable application should check for both possibilities.

EBADF
fd is not a valid file descriptor or is not open for writing.

EFAULT
buf is outside your accessible address space.

EFBIG
An attempt was made to write a file that exceeds the implementation-defined maximum file size or the process's file size limit, or to write at a position past the maximum allowed offset.

EINTR
The call was interrupted by a signal before any data was written; see **signal**(7).

EINVAL
fd is attached to an object which is unsuitable for writing; or the file was opened with the **O_DIRECT** flag, and either the address specified in *buf*, the value specified in *count*, or the current file offset is not suitably aligned.

EIO A low-level I/O error occurred while modifying the inode.

ENOSPC
The device containing the file referred to by *fd* has no room for the data.

EPIPE *fd* is connected to a pipe or socket whose reading end is closed. When this happens the writing process will also receive a **SIGPIPE** signal. (Thus, the write return value is seen only if the program catches, blocks or ignores this signal.)

Other errors may occur, depending on the object connected to *fd*.

CONFORMING TO

SVr4, 4.3BSD, POSIX.1-2001.

Under SVr4 a write may be interrupted and return **EINTR** at any point, not just before any data is written.

NOTES

A successful return from **write**() does not make any guarantee that data has been committed to disk. In fact, on some buggy implementations, it does not even guarantee that space has successfully been reserved for the data. The only way to be sure is to call **fsync**(2) after you are done writing all your data.

If a **write**() is interrupted by a signal handler before any bytes are written, then the call fails with the error **EINTR**; if it is interrupted after at least one byte has been written, the call succeeds, and returns the number of bytes written.

SEE ALSO

close(2), **fcntl**(2), **fsync**(2), **ioctl**(2), **lseek**(2), **open**(2), **pwrite**(2), **read**(2), **select**(2), **writev**(2), **fwrite**(3)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.