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 to the file referenced by the file descriptor *fd* from the buffer starting at *buf*. POSIX requires that a **read()** 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 are returned (zero indicates nothing was written). On error, −1 is returned, and *errno* is set appropriately. If *count* is zero and the file descriptor refers to a regular file, 0 will be returned without causing any other effect. For a special file, the results are not portable.

ERRORS

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

EINVAL
fd is attached to an object which is unsuitable 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' file size limit, or to write at a position past than the maximum allowed offset.

EPIPE *fd* is connected to a pipe or socket whose reading end is closed. When this happens the writing process will receive a **SIGPIPE** signal; if it catches, blocks or ignores this the error **EPIPE** is returned.

EAGAIN
Non-blocking I/O has been selected using **O_NONBLOCK** and the write would block.

EINTR
The call was interrupted by a signal before any data was written.

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

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

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

CONFORMING TO

SVr4, SVID, POSIX, X/OPEN, 4.3BSD. SVr4 documents additional error conditions EDEADLK, ENOLCK, ENOLNK, ENOSR, ENXIO, EPIPE, or ERANGE. 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.

SEE ALSO

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