NAME

write, **writev**, **pwrite**, **pwritev** - write output

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

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

ssize_t
pwrite(*int fd*, *const void *buf*, *size_t nbytes*, *off_t offset*);

#include <sys/uio.h>

ssize_t
writev(*int fd*, *const struct iovec *iov*, *int iovcnt*);

ssize_t
pwritev(*int fd*, *const struct iovec *iov*, *int iovcnt*, *off_t offset*);

DESCRIPTION

The **write**() system call attempts to write *nbytes* of data to the object referenced by the descriptor *fd* from the buffer pointed to by *buf*. The **writev**() system call performs the same action, but gathers the output data from the *iovcnt* buffers specified by the members of the *iov* array: iov[0], iov[1], ..., iov[iovcnt-1]. The **pwrite**() and **pwritev**() system calls perform the same functions, but write to the specified position in the file without modifying the file pointer.

For **writev**() and **pwritev**(), the *iovec* structure is defined as:

```
struct iovec {
        void   *iov_base;  /* Base address. */
        size_t iov_len;    /* Length. */
};
```

Each *iovec* entry specifies the base address and length of an area in memory from which data should be written. The **writev**() system call will always write a complete area before proceeding to the next.

On objects capable of seeking, the **write**() starts at a position given by the pointer associated with *fd*, see lseek(2). Upon return from **write**(), the pointer is incremented by the number of bytes which were written.

Objects that are not capable of seeking always write from the current position. The value of the pointer associated with such an object is undefined.

If the real user is not the super-user, then **write**() clears the set-user-id bit on a file. This prevents penetration of system security by a user who "captures" a writable set-user-id file owned by the super-user.

When using non-blocking I/O on objects such as sockets that are subject to flow control, **write**() and **writev**() may write fewer bytes than requested; the return value must be noted, and the remainder of the operation should be retried when possible.

RETURN VALUES

Upon successful completion the number of bytes which were written is returned. Otherwise a -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **write**(), **writev**(), **pwrite**() and **pwritev**() system calls will fail and the file pointer will remain unchanged if:

[EBADF]	The *fd* argument is not a valid descriptor open for writing.
[EPIPE]	An attempt is made to write to a pipe that is not open for reading by any process.
[EPIPE]	An attempt is made to write to a socket of type SOCK_STREAM that is not connected to a peer socket.
[EFBIG]	An attempt was made to write a file that exceeds the process's file size limit or the maximum file size.
[EFAULT]	Part of *iov* or data to be written to the file points outside the process's allocated address space.
[EINVAL]	The pointer associated with *fd* was negative.
[ENOSPC]	There is no free space remaining on the file system containing the file.

[EDQUOT]	The user’s quota of disk blocks on the file system containing the file has been exhausted.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[EINTR]	A signal interrupted the write before it could be completed.
[EAGAIN]	The file was marked for non-blocking I/O, and no data could be written immediately.
[EROFS]	An attempt was made to write over a disk label area at the beginning of a slice. Use disklabel(8) **-W** to enable writing on the disk label area.
[EINVAL]	The value *nbytes* is greater than SSIZE_MAX (or greater than INT_MAX, if the sysctl *debug.iosize_max_clamp* is non-zero).
[EINTEGRITY]	The backing store for *fd* detected corrupted data while reading. (For example, writing a partial filesystem block may require first reading the existing block which may trigger this error.)

In addition, **writev**() and **pwritev**() may return one of the following errors:

[EDESTADDRREQ]	The destination is no longer available when writing to a UNIX domain datagram socket on which connect(2) had been used to set a destination address.
[EINVAL]	The *iovcnt* argument was less than or equal to 0, or greater than IOV_MAX.
[EINVAL]	One of the *iov_len* values in the *iov* array was negative.
[EINVAL]	The sum of the *iov_len* values is greater than SSIZE_MAX (or greater than INT_MAX, if the sysctl *debug.iosize_max_clamp* is non-zero).
[ENOBUFS]	The mbuf pool has been completely exhausted when writing to a socket.

The **pwrite**() and **pwritev**() system calls may also return the following errors:

[EINVAL]	The *offset* value was negative.
[ESPIPE]	The file descriptor is associated with a pipe, socket, or FIFO.

SEE ALSO

fcntl(2), lseek(2), open(2), pipe(2), select(2)

STANDARDS

The **write**() system call is expected to conform to IEEE Std 1003.1-1990 ("POSIX.1"). The **writev**() and **pwrite**() system calls are expected to conform to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2").

HISTORY

The **pwritev**() system call appeared in FreeBSD 6.0. The **pwrite**() function appeared in AT&T System V Release 4 UNIX. The **writev**() system call appeared in 4.2BSD. The **write**() function appeared in Version 1 AT&T UNIX.

BUGS

The **pwrite**() system call appends the file without changing the file offset if O_APPEND is set, contrary to IEEE Std 1003.1-2008 ("POSIX.1") where **pwrite**() writes into *offset* regardless of whether O_APPEND is set.