System Calls write(2)

NAME

write, pwrite, writev - write on a file

SYNOPSIS

```
#include <unistd.h>

ssize_t write(int fildes, const void *buf, size_t nbyte);

ssize_t pwrite(int fildes, const void  *buf,  size_t  nbyte,
off_t offset);

#include <sys/uio.h>

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

DESCRIPTION

The write() function attempts to write nbyte bytes from the buffer pointed to by buf to the file associated with the open file descriptor, fildes.

If nbyte is 0, write() will return 0 and have no other results if the file is a regular file; otherwise, the results are unspecified.

On a regular file or other file capable of seeking, the actual writing of data proceeds from the position in the file indicated by the file offset associated with fildes. Before successful return from write(), the file offset is incremented by the number of bytes actually written. On a regular file, if this incremented file offset is greater than the length of the file, the length of the file will be set to this file offset.

If the O_SYNC bit has been set, write I/O operations on the file descriptor complete as defined by synchronized I/O file integrity completion.

If fildes refers to a socket, write() is equivalent to send(3SOCKET) with no flags set.

On a file not capable of seeking, writing always takes place starting at the current position. The value of a file offset associated with such a device is undefined.

If the O_APPEND flag of the file status flags is set, the file offset will be set to the end of the file prior to each write and no intervening file modification operation will occur between changing the file offset and the write operation.

For regular files, no data transfer will occur past the offset maximum established in the open file description with fildes.

A write() to a regular file is blocked if mandatory file/record locking is set (see chmod(2)), and there is a record lock owned by another process on the segment of the file to be written:

- If O_NDELAY or O_NONBLOCK is set, write() returns -1 and sets errno to EAGAIN.
- If O_NDELAY and O_NONBLOCK are clear, write() sleeps until all blocking locks are removed or the write() is terminated by a signal.

If a write() requests that more bytes be written than there is room for-for example, if the write would exceed the process file size limit (see getrlimit(2) and ulimit(2)), the system file size limit, or the free space on the device-only as many bytes as there is room for will be written. For example, suppose there is space for 20 bytes more in a file before reaching a limit. A write() of 512-bytes returns 20. The next write() of a non-zero number of bytes gives a failure return (except as noted for pipes and FIFO below).

If write() is interrupted by a signal before it writes any data, it will return -1 with errno set to EINTR.

If write() is interrupted by a signal after it successfully writes some data, it will return the number of bytes written.

After a write() to a regular file has successfully returned:

- Any successful read(2) from each byte position in the file that was modified by that write will return the data specified by the write() for that position until such byte positions are again modified.

- Any subsequent successful write() to the same byte position in the file will overwrite that file data.

Write requests to a pipe or FIFO are handled the same as a regular file with the following exceptions:

- There is no file offset associated with a pipe, hence each write request appends to the end of the pipe.

- Write requests of {PIPE_BUF} bytes or less are guaranteed not to be interleaved with data from other processes doing writes on the same pipe. Writes of greater than {PIPE_BUF} bytes may have data interleaved, on arbitrary boundaries, with writes by other processes, whether or not the O_NONBLOCK or O_NDELAY flags are set.

- If O_NONBLOCK and O_NDELAY are clear, a write request may cause the process to block, but on normal completion it returns nbyte.

- If O_NONBLOCK and O_NDELAY are set, write() does not block the process. If a write() request for PIPE_BUF or fewer bytes succeeds completely write() returns nbyte. Otherwise, if O_NONBLOCK is set, it returns -1 and sets errno to EAGAIN or if O_NDELAY is set, it returns 0. A write() request for greater than {PIPE_BUF} bytes transfers what it can and returns the number of bytes written or it transfers no data and, if O_NONBLOCK is set, returns -1 with errno set to EAGAIN or if O_NDELAY is set, it returns 0. Finally, if a request is greater than PIPE_BUF bytes and all data previously written to the pipe has been read, write() transfers at least PIPE_BUF bytes.

When attempting to write to a file descriptor (other than a pipe, a FIFO, a socket, or a STREAM) that supports nonblocking writes and cannot accept the data immediately:

- If O_NONBLOCK and O_NDELAY are clear, write() blocks until the data can be accepted.

- If O_NONBLOCK or O_NDELAY is set, write() does not block the process. If some data can be written without blocking the process, write() writes what it can and returns the number of bytes written. Otherwise, if O_NONBLOCK is set, it returns -1 and sets errno to EAGAIN or if O_NDELAY is set, it returns 0.

Upon successful completion, where nbyte is greater than 0, write() will mark for update the st_ctime and st_mtime fields of the file, and if the file is a regular file, the S_ISUID and S_ISGID bits of the file mode may be cleared.

For STREAMS files (see intro(2) and streamio(7I)), the operation of write() is determined by the values of the minimum and maximum nbyte range ("packet size") accepted by the STREAM. These values are contained in the topmost STREAM module, and can not be set or tested from user level. If nbyte falls within the packet size range, nbyte bytes are written. If nbyte does not fall within the range and the minimum packet size value is zero, write() breaks the buffer into maximum packet size segments prior to sending the data downstream (the last segment may be smaller than the maximum packet size). If nbyte does not fall within the range and the minimum value is non-zero, write() fails and

sets errno to ERANGE. Writing a zero-length buffer (nbyte is zero) to a STREAMS device sends a zero length message with zero returned. However, writing a zero-length buffer to a pipe or FIFO sends no message and zero is returned. The user program may issue the I_SWROPT ioctl(2) to enable zero-length messages to be sent across the pipe or FIFO (see streamio(7I)).

When writing to a STREAM, data messages are created with a priority band of zero. When writing to a socket or to a STREAM that is not a pipe or a FIFO:

- If O_NDELAY and O_NONBLOCK are not set, and the STREAM cannot accept data (the STREAM write queue is full due to internal flow control conditions), write() blocks until data can be accepted.
- If O_NDELAY or O_NONBLOCK is set and the STREAM cannot accept data, write() returns -1 and sets errno to EAGAIN.
- If O_NDELAY or O_NONBLOCK is set and part of the buffer has already been written when a condition occurs in which the STREAM cannot accept additional data, write() terminates and returns the number of bytes written.

The write() and writev() functions will fail if the STREAM head had processed an asynchronous error before the call. In this case, the value of errno does not reflect the result of write() or writev() but reflects the prior error.

pwrite()
The pwrite() function performs the same action as write(), except that it writes into a given position without changing the file pointer. The first three arguments to pwrite() are the same as write() with the addition of a fourth argument offset for the desired position inside the file.

writev()
The writev() function performs the same action as write(), 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 iovcnt buffer is valid if greater than 0 and less than or equal to {IOV_MAX}. See intro(2) for a definition of {IOV_MAX}.

The iovec structure contains the following members:

```
caddr_t  iov_base;
int      iov_len;
```

Each iovec entry specifies the base address and length of an area in memory from which data should be written. The writev() function always writes all data from an area before proceeding to the next.

If fildes refers to a regular file and all of the iov_len members in the array pointed to by iov are 0, writev() will return 0 and have no other effect. For other file types, the behavior is unspecified.

If the sum of the iov_len values is greater than SSIZE_MAX, the operation fails and no data is transferred.

RETURN VALUES
Upon successful completion, write() returns the number of bytes actually written to the file associated with fildes. This number is never greater than nbyte. Otherwise, -1 is returned, the file-pointer remains unchanged, and errno is set to indicate the error.

Upon successful completion, writev() returns the number of bytes actually written. Otherwise, it returns -1, the file-pointer remains unchanged, and errno is set to indicate an error.

ERRORS

The write(), pwrite(), and writev() functions will fail if:

EAGAIN
Mandatory file/record locking is set, O_NDELAY or O_NONBLOCK is set, and there is a blocking record lock; an attempt is made to write to a STREAM that can not accept data with the O_NDELAY or O_NONBLOCK flag set; or a write to a pipe or FIFO of PIPE_BUF bytes or less is requested and less than nbytes of free space is available.

EBADF The fildes argument is not a valid file descriptor open for writing.

EDEADLK
The write was going to go to sleep and cause a deadlock situation to occur.

EDQUOT
The user's quota of disk blocks on the file system containing the file has been exhausted.

EFBIG An attempt is made to write a file that exceeds the process's file size limit or the maximum file size (see getrlimit(2) and ulimit(2)).

EFBIG The file is a regular file, nbyte is greater than 0, and the starting position is greater than or equal to the offset maximum established in the file description associated with fildes.

EINTR A signal was caught during the write operation and no data was transferred.

EIO The process is in the background and is attempting to write to its controlling terminal whose TOSTOP flag is set, or the process is neither ignoring nor blocking SIGTTOU signals and the process group of the process is orphaned.

ENOLCK
Enforced record locking was enabled and {LOCK_MAX} regions are already locked in the system, or the system record lock table was full and the write could not go to sleep until the blocking record lock was removed.

ENOLINK
The fildes argument is on a remote machine and the link to that machine is no longer active.

ENOSPC
During a write to an ordinary file, there is no free space left on the device.

ENOSR An attempt is made to write to a STREAMS with insufficient STREAMS memory resources available in the system.

ENXIO A hangup occurred on the STREAM being written to.

EPIPE An attempt is made to write to a pipe or a FIFO that is not open for reading by any process, or that has only one end open (or to a file descriptor created by socket(3SOCKET), using type SOCK_STREAM that is no longer connected to a peer endpoint). A SIGPIPE signal will also be sent to the process. The process dies unless special provisions were taken to catch or ignore the signal.

ERANGE
The transfer request size was outside the range supported by the STREAMS file associated with fildes.

The write() and pwrite() functions will fail if:

EFAULT
The buf argument points to an illegal address.

EINVAL
The nbyte argument overflowed an ssize_t.

The pwrite() function fails and the file pointer remains unchanged if:

ESPIPE
The fildes argument is associated with a pipe or FIFO.

The write() and writev() functions may fail if:

EINVAL
The STREAM or multiplexer referenced by fildes is linked (directly or indirectly) downstream from a multiplexer.

ENXIO A request was made of a non-existent device, or the request was outside the capabilities of the device.

ENXIO A hangup occurred on the STREAM being written to.

A write to a STREAMS file may fail if an error message has been received at the STREAM head. In this case, errno is set to the value included in the error message.

The writev() function may fail if:

EINVAL
The iovcnt argument was less than or equal to 0 or greater than {IOV_MAX}; one of the iov_len values in the iov array was negative; or the sum of the iov_len values in the iov array overflowed an ssize_t.

USAGE
The pwrite() function has a transitional interface for 64-bit file offsets. See lf64(5).

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard
MT-Level	write() is Async-Signal-Safe

SEE ALSO
intro(2), chmod(2), creat(2), dup(2), fcntl(2), getrlimit(2), ioctl(2), lseek(2), open(2), pipe(2), ulimit(2), send(3SOCKET), socket(3SOCKET), attributes(5), lf64(5), streamio(7I)

SunOS 5.9 Last change: 18 Oct 2001