NAME

read, **readv**, **pread**, **preadv** - read input

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

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

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

#include <sys/uio.h>

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

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

DESCRIPTION

The **read**() system call attempts to read *nbytes* of data from the object referenced by the descriptor *fd* into the buffer pointed to by *buf*. The **readv**() system call performs the same action, but scatters the input data into the *iovcnt* buffers specified by the members of the *iov* array: iov[0], iov[1], ..., iov[iovcnt-1]. The **pread**() and **preadv**() system calls perform the same functions, but read from the specified position in the file without modifying the file pointer.

For **readv**() and **preadv**(), 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 where data should be placed. The **readv**() system call will always fill an area completely before proceeding to the next.

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

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

Upon successful completion, **read**(), **readv**(), **pread**() and **preadv**() return the number of bytes actually read and placed in the buffer. The system guarantees to read the number of bytes requested if the descriptor references a normal file that has that many bytes left before the end-of-file, but in no other case.

In accordance with IEEE Std 1003.1-2004 ("POSIX.1"), both read(2) and write(2) syscalls are atomic with respect to each other in the effects on file content, when they operate on regular files. If two threads each call one of the read(2) or write(2), syscalls, each call will see either all of the changes of the other call, or none of them. The FreeBSD kernel implements this guarantee by locking the file ranges affected by the calls.

RETURN VALUES

If successful, the number of bytes actually read is returned. Upon reading end-of-file, zero is returned. Otherwise, a -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **read**(), **readv**(), **pread**() and **preadv**() system calls will succeed unless:

[EBADF]	The *fd* argument is not a valid file or socket descriptor open for reading.
[ECONNRESET]	The *fd* argument refers to a socket, and the remote socket end is forcibly closed.
[EFAULT]	The *buf* argument points outside the allocated address space.
[EIO]	An I/O error occurred while reading from the file system.
[EINTEGRITY]	Corrupted data was detected while reading from the file system.
[EBUSY]	Failed to read from a file, e.g. /proc/<pid>/regs while <pid> is not stopped
[EINTR]	A read from a slow device (i.e. one that might block for an arbitrary amount of time) was interrupted by the delivery of a signal before any data arrived.
[EINVAL]	The pointer associated with *fd* was negative.

[EAGAIN] The file was marked for non-blocking I/O, and no data were ready to be read.

[EISDIR] The file descriptor is associated with a directory. Directories may only be read directly by root if the filesystem supports it and the security.bsd.allow_read_dir sysctl MIB is set to a non-zero value. For most scenarios, the readdir(3) function should be used instead.

[EOPNOTSUPP] The file descriptor is associated with a file system and file type that do not allow regular read operations on it.

[EOVERFLOW] The file descriptor is associated with a regular file, *nbytes* is greater than 0, *offset* is before the end-of-file, and *offset* is greater than or equal to the offset maximum established for this file system.

[EINVAL] The value *nbytes* is greater than INT_MAX.

In addition, **readv**() and **preadv**() may return one of the following errors:

[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 in the *iov* array overflowed a 32-bit integer.

[EFAULT] Part of the *iov* array points outside the process’s allocated address space.

The **pread**() and **preadv**() 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

dup(2), fcntl(2), getdirentries(2), open(2), pipe(2), select(2), socket(2), socketpair(2), fread(3), readdir(3)

STANDARDS

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

HISTORY

The **preadv**() system call appeared in FreeBSD 6.0. The **pread**() function appeared in AT&T System V Release 4 UNIX. The **readv**() system call appeared in 4.2BSD. The **read**() function appeared in Version 1 AT&T UNIX.