NAME

readv, writev, preadv, pwritev − read or write data into multiple buffers

SYNOPSIS

```
#include <sys/uio.h>

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

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

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

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

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

preadv(), **pwritev**(): _BSD_SOURCE

DESCRIPTION

The **readv**() system call reads *iovcnt* buffers from the file associated with the file descriptor *fd* into the buffers described by *iov* ("scatter input").

The **writev**() system call writes *iovcnt* buffers of data described by *iov* to the file associated with the file descriptor *fd* ("gather output").

The pointer *iov* points to an array of *iovec* structures, defined in <*sys/uio.h*> as:

```
struct iovec {
    void  *iov_base;    /* Starting address */
    size_t iov_len;     /* Number of bytes to transfer */
};
```

The **readv**() system call works just like **read**(2) except that multiple buffers are filled.

The **writev**() system call works just like **write**(2) except that multiple buffers are written out.

Buffers are processed in array order. This means that **readv**() completely fills *iov*[0] before proceeding to *iov*[1], and so on. (If there is insufficient data, then not all buffers pointed to by *iov* may be filled.) Similarly, **writev**() writes out the entire contents of *iov*[0] before proceeding to *iov*[1], and so on.

The data transfers performed by **readv**() and **writev**() are atomic: the data written by **writev**() is written as a single block that is not intermingled with output from writes in other processes (but see **pipe**(7) for an exception); analogously, **readv**() is guaranteed to read a contiguous block of data from the file, regardless of read operations performed in other threads or processes that have file descriptors referring to the same open file description (see **open**(2)).

preadv() and pwritev()

The **preadv**() system call combines the functionality of **readv**() and **pread**(2). It performs the same task as **readv**(), but adds a fourth argument, *offset*, which specifies the file offset at which the input operation is to be performed.

The **pwritev**() system call combines the functionality of **writev**() and **pwrite**(2). It performs the same task as **writev**(), but adds a fourth argument, *offset*, which specifies the file offset at which the output operation is to be performed.

The file offset is not changed by these system calls. The file referred to by *fd* must be capable of seeking.

RETURN VALUE

On success, **readv**() and **preadv**() return the number of bytes read; **writev**() and **pwritev**() return the number of bytes written. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

The errors are as given for **read**(2) and **write**(2). Furthermore, **preadv**() and **pwritev**() can also fail for the same reasons as **lseek**(2). Additionally, the following error is defined:

EINVAL

The sum of the *iov_len* values overflows an *ssize_t* value. Or, the vector count *iovcnt* is less than zero or greater than the permitted maximum.

VERSIONS

preadv() and **pwritev**() first appeared in Linux 2.6.30; library support was added in glibc 2.10.

CONFORMING TO

readv(), **writev**(): 4.4BSD (these system calls first appeared in 4.2BSD), POSIX.1-2001. Linux libc5 used *size_t* as the type of the *iovcnt* argument, and *int* as the return type.

preadv(), **pwritev**(): nonstandard, but present also on the modern BSDs.

NOTES

Linux Notes

POSIX.1-2001 allows an implementation to place a limit on the number of items that can be passed in *iov*. An implementation can advertise its limit by defining **IOV_MAX** in <*limits.h*> or at run time via the return value from *sysconf(_SC_IOV_MAX)*. On Linux, the limit advertised by these mechanisms is 1024, which is the true kernel limit. However, the glibc wrapper functions do some extra work if they detect that the underlying kernel system call failed because this limit was exceeded. In the case of **readv**() the wrapper function allocates a temporary buffer large enough for all of the items specified by *iov*, passes that buffer in a call to **read**(2), copies data from the buffer to the locations specified by the *iov_base* fields of the elements of *iov*, and then frees the buffer. The wrapper function for **writev**() performs the analogous task using a temporary buffer and a call to **write**(2).

BUGS

It is not advisable to mix calls to **readv**() or **writev**(), which operate on file descriptors, with the functions from the stdio library; the results will be undefined and probably not what you want.

EXAMPLE

The following code sample demonstrates the use of **writev**():

```
char *str0 = "hello ";
char *str1 = "world\n";
struct iovec iov[2];
ssize_t nwritten;

iov[0].iov_base = str0;
iov[0].iov_len = strlen(str0);
iov[1].iov_base = str1;
iov[1].iov_len = strlen(str1);

nwritten = writev(STDOUT_FILENO, iov, 2);
```

SEE ALSO

pread(2), **read**(2), **write**(2)

COLOPHON

This page is part of release 3.32 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/.