Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'pwritev.2' command

$ man pwritev.2

READV(2) Linux Programmer's Manual READV(2)

NAME

readv, writev, preadv, pwritev, preadv2, pwritev2 - 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);

ssize_t preadv2(int fd, const struct iovec *iov, int iovcnt,
                off_t offset, int flags);

ssize_t pwritev2(int fd, const struct iovec *iov, int iovcnt,
                 off_t offset, int flags);
```

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

preadv(), pwritev():

Since glibc 2.19:

_DEFAULT_SOURCE

Glibc 2.19 and earlier:

_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 re? ferred to by fd must be capable of seeking.

preadv2() and pwritev2()

These system calls are similar to preadv() and pwritev() calls, but add a fifth argument, flags, which modifies the behavior on a per-call ba? sis.

Unlike preadv() and pwritev(), if the offset argument is -1, then the current file offset is used and updated.

The flags argument contains a bitwise OR of zero or more of the follow? ing flags:

RWF_DSYNC (since Linux 4.7)

Provide a per-write equivalent of the O_DSYNC open(2) flag. This flag is meaningful only for pwritev2(), and its effect ap? plies only to the data range written by the system call.

RWF_HIPRI (since Linux 4.6)

High priority read/write. Allows block-based filesystems to use polling of the device, which provides lower latency, but may use additional resources. (Currently, this feature is usable only on a file descriptor opened using the O_DIRECT flag.)

RWF_SYNC (since Linux 4.7)

Provide a per-write equivalent of the O_SYNC open(2) flag. This flag is meaningful only for pwritev2(), and its effect applies only to the data range written by the system call.

RWF_NOWAIT (since Linux 4.14)

Do not wait for data which is not immediately available. If this flag is specified, the preadv2() system call will return instantly if it would have to read data from the backing storage or wait for a lock. If some data was successfully read, it will return the number of bytes read. If no bytes were read, it will return -1 and set errno to EAGAIN. Currently, this flag is

meaningful only for preadv2().

RWF_APPEND (since Linux 4.16)

Provide a per-write equivalent of the O_APPEND open(2) flag. This flag is meaningful only for pwritev2(), and its effect ap? plies only to the data range written by the system call. The offset argument does not affect the write operation; the data is always appended to the end of the file. However, if the offset argument is -1, the current file offset is updated.

RETURN VALUE

On success, readv(), preadv(), and preadv2() return the number of bytes read; writev(), pwritev(), and pwritev2() return the number of bytes written.

Note that it is not an error for a successful call to transfer fewer bytes than requested (see read(2) and write(2)).

On error, -1 is returned, and errno is set appropriately.

ERRORS

The errors are as given for read(2) and write(2). Furthermore, preadv(), preadv2(), pwritev(), and pwritev2() can also fail for the same reasons as lseek(2). Additionally, the following errors are de? fined:

EINVAL The sum of the iov_len values overflows an ssize_t value.

EINVAL The vector count, iovcnt, is less than zero or greater than the permitted maximum.

EOPNOTSUPP

An unknown flag is specified in flags.

VERSIONS

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

preadv2() and pwritev2() first appeared in Linux 4.6. Library support was added in glibc 2.26.

CONFORMING TO

readv(), writev(): POSIX.1-2001, POSIX.1-2008, 4.4BSD (these system calls first appeared in 4.2BSD).

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

preadv2(), pwritev2(): nonstandard Linux extension.

NOTES

POSIX.1 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 modern Linux systems, the limit is 1024. Back in Linux 2.0 days, this limit was 16.

C library/kernel differences

The raw preadv() and pwritev() system calls have call signatures that differ slightly from that of the corresponding GNU C library wrapper functions shown in the SYNOPSIS. The final argument, offset, is un? packed by the wrapper functions into two arguments in the system calls:

unsigned long pos_l, unsigned long pos

These arguments contain, respectively, the low order and high order 32 bits of offset.

Historical C library/kernel differences

To deal with the fact that IOV_MAX was so low on early versions of Linux, the glibc wrapper functions for readv() and writev() did some extra work if they detected that the underlying kernel system call failed because this limit was exceeded. In the case of readv(), the wrapper function allocated a temporary buffer large enough for all of the items specified by iov, passed that buffer in a call to read(2), copied data from the buffer to the locations specified by the iov_base fields of the elements of iov, and then freed the buffer. The wrapper function for writev() performed the analogous task using a temporary buffer and a call to write(2).

The need for this extra effort in the glibc wrapper functions went away with Linux 2.2 and later. However, glibc continued to provide this be? havior until version 2.10. Starting with glibc version 2.9, the wrap? per functions provide this behavior only if the library detects that the system is running a Linux kernel older than version 2.6.18 (an ar? bitrarily selected kernel version). And since glibc 2.20 (which re?

quires a minimum Linux kernel version of 2.6.32), the glibc wrapper functions always just directly invoke the system calls.

EXAMPLES

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 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.