NAME

lio_listio – list directed I/O

SYNOPSIS

cc [*flag* ...] *file* ... **–lposix4** [*library* ...]

#include <aio.h>

int lio_listio(int *mode***, struct aiocb** ∗ **const** *list***[ ], int** *nent***, struct sigevent** ∗*sig***);**

```
struct aiocb {
  int                aio_fildes;      /* file descriptor */
  volatile void      *aio_buf;        /* buffer location */
  size_t             aio_nbytes;      /* length of transfer */
  off_t              aio_offset;      /* file offset */
  int                aio_reqprio;     /* request priority offset */
  struct sigevent    aio_sigevent;    /* signal number and offset */
  int                aio_lio_opcode;  /* listio operation */
};
struct sigevent {
  int                sigev_notify;    /* notification mode */
  int                sigev_signo;     /* signal number */
  union sigval       sigev_value;     /* signal value */
};
union sigval {
  int                sival_int;       /* integer value */
  void               *sival_ptr;      /* pointer value */
};
```

MT-LEVEL

MT-Safe

DESCRIPTION

lio_listio( ) allows the calling process, LWP, or thread, to initiate a list of I/O requests within a single function call.

If *mode* is set to **LIO_WAIT, lio_listio( )** behaves synchronously, waiting until all I/O is completed, and the *sig* argument is ignored. If *mode* is set to **LIO_NOWAIT, lio_listio( )** behaves asynchronously; returning immediately, and signal delivery will occur, according to the *sig* argument, when all the I/O operations from this function complete. If *sig* is **NULL,** or the **sigev_signo** member of the **sigevent** structure referenced by *sig* is zero, then no signal delivery will occur. Otherwise, the signal number indicated by **sigev_signo** will be delivered when all the requests in *list* have completed.

list is an array of pointers to **aiocb** structures. This array consists of *nent* elements. The array may contain **NULL** pointers, which will be ignored.

The **aio_lio_opcode** field of each **aiocb** structure in *list* specifies the operation to be performed (see **/usr/include/aio.h**).

LIO_READ
requests **aio_read**(3R).

LIO_WRITE
requests **aio_write**(3R).

LIO_NOP
causes the *list* entry to be ignored.

nent specifies the length of the array (number of members of the list).

When *mode* has the value **LIO_WAIT**, a pointer to a signal control structure, *sig,* is used to define both the signal to be generated and how the calling process will be notified upon I/O completion. If *sig*->**sigev_notify** is **SIGEV_NONE**, then no signal will be posted upon I/O completion, but the error status and the return status for the operation will be set appropriately. If *sig*->**sigev_notify** is **SIGEV_SIGNAL**, then the signal specified in *sig*->**sigev_signo** will be sent to the process. If the **SA_SIGINFO** flag is set for that signal number, then the signal will be queued to the process and the value specified in *sig*->**sigev_value** will be the **si_value** component of the generated signal (see **siginfo**(5)).

The behavior of this function is altered according to the definitions of synchronized I/O data integrity completion and synchronized I/O file integrity completion if synchronized I/O is enabled on the file associated with **aio_fildes.** (see **fcntl**(5) definitions of **O_DSYNC** and **O_SYNC**.)

RETURN VALUES

If the *mode* argument has the value **LIO_NOWAIT**, and the I/O operations are successfully queued, **lio_listio( )** returns **0;** otherwise, it returns **-1,** and sets **errno** to indicate the error condition.

If the *mode* argument has the value **LIO_WAIT**, and when all the indicated I/O has completed successfully, **lio_listio( )** returns **0;** otherwise, it returns **-1,** and sets **errno** to indicate the error condition.

In either case, the return value only indicates the success or failure of the **lio_listio( )** call itself, not the status of the individual I/O requests. In some cases, one or more of the I/O requests contained in the list may fail. Failure of an individual request does not prevent completion of any other individual request. To determine the outcome of each I/O request, the application must examine the error status associated with each *aiocb* control block. Each error status so returned is identical to that returned as a result of an **aio_read**(3R) or **aio_write**(3R) function.

ERRORS

EAGAIN The resources necessary to queue all the I/O requests were not available. The error status for each request is recorded in the **aio_error** member of the corresponding **aiocb** structure, and can be retrieved using **aio_error**(3R).

nent entries exceed the system-wide limit, **AIO_MAX**.

EINVAL The *mode* argument is an improper value.

The value of *nent* is greater than **AIO_LISTIO_MAX**.

EINTR A signal was delivered while waiting for all I/O requests to complete during an **LIO_WAIT** operation. However, the outstanding I/O requests are not canceled. Use **aio_fsync**(3R) to determine if any request was initiated; **aio_return**(3R) to determine if any request has completed; or **aio_error**(3R) to determine if any request was canceled.

EIO One or more of the individual I/O operations failed. Using **aio_error**(3R) with each **aiocb** structure will determine the individual request(s) that failed.

ENOSYS **lio_listio( )** is not supported by this implementation.

If either **lio_listio( )** succeeds in queuing all of its requests, or **errno** is set to **EAGAIN**, **EINTR**, or **EIO**, then some of the I/O specified from the list may have been initiated. In this event, each **aiocb** structure contains errors specific to the **read**(2) or **write**(2) function being performed; i.e.:

EAGAIN The requested I/O operation was not queued due to resource limitations.

ECANCELED The requested I/O was canceled before the I/O completed due to an explicit **aio_cancel**(3R) request.

EINPROGRESS Thee requested I/O is in progress.

SEE ALSO

close(2), **exec**(2), **exit**(2), **fork**(2), **lseek**(2), **read**(2), **write**(2), **aio_cancel**(3R), **aio_read**(3R), **aio_return**(3R), **fcntl**(5), **siginfo**(5)

NOTES

Applications compiled under Solaris 2.3 and 2.4 and using POSIX aio must be recompiled to work correctly when Solaris supports the Asynchronous Input and Output option.

BUGS

In Solaris 2.5, these functions always return –**1** and set **errno** to **ENOSYS,** because this release does not support the Asynchronous Input and Output option. It is our intention to provide support for these interfaces in future releases.