Red Hat Enterprise Linux Release 9.2 Manual Pages on 'lio_listio.3' command

$ man lio_listio.3

LIO_LISTIO(3)   Linux Programmer's Manual   LIO_LISTIO(3)

NAME

lio_listio - initiate a list of I/O requests

SYNOPSIS

```
#include <aio.h>

int lio_listio(int mode, struct aiocb *const aiocb_list[],

               int nitems, struct sigevent *sevp);

Link with -lrt.
```

DESCRIPTION

The lio_listio() function initiates the list of I/O operations de?

scribed by the array aiocb_list.

The mode operation has one of the following values:

LIO_WAIT

The call blocks until all operations are complete. The sevp ar?

gument is ignored.

LIO_NOWAIT

The I/O operations are queued for processing and the call re?

turns immediately. When all of the I/O operations complete,

asynchronous notification occurs, as specified by the sevp argu?

ment; see sigevent(7) for details. If sevp is NULL, no asyn?

chronous notification occurs.

The aiocb_list argument is an array of pointers to aiocb structures

that describe I/O operations. These operations are executed in an un?

specified order. The nitems argument specifies the size of the array aiocb_list. null pointers in aiocb_list are ignored.

In each control block in aiocb_list, the aio_lio_opcode field specifies the I/O operation to be initiated, as follows:

LIO_READ

Initiate a read operation. The operation is queued as for a call to aio_read(3) specifying this control block.

LIO_WRITE

Initiate a write operation. The operation is queued as for a call to aio_write(3) specifying this control block.

LIO_NOP

Ignore this control block.

The remaining fields in each control block have the same meanings as for aio_read(3) and aio_write(3). The aio_sigevent fields of each con?trol block can be used to specify notifications for the individual I/O operations (see sigevent(7)).

RETURN VALUE

If mode is LIO_NOWAIT, lio_listio() returns 0 if all I/O operations are successfully queued. Otherwise, -1 is returned, and errno is set to indicate the error.

If mode is LIO_WAIT, lio_listio() returns 0 when all of the I/O opera?tions have completed successfully. Otherwise, -1 is returned, and er?rno is set to indicate the error.

The return status from lio_listio() provides information only about the call itself, not about the individual I/O operations. One or more of the I/O operations may fail, but this does not prevent other operations completing. The status of individual I/O operations in aiocb_list can be determined using aio_error(3). When an operation has completed, its return status can be obtained using aio_return(3). Individual I/O op?erations can fail for the reasons described in aio_read(3) and aio_write(3).

ERRORS

The lio_listio() function may fail for the following reasons:

EAGAIN Out of resources.

EAGAIN The number of I/O operations specified by nitems would cause the
limit AIO_MAX to be exceeded.

EINTR mode was LIO_WAIT and a signal was caught before all I/O opera?
tions completed; see signal(7). (This may even be one of the
signals used for asynchronous I/O completion notification.)

EINVAL mode is invalid, or nitems exceeds the limit AIO_LISTIO_MAX.

EIO One of more of the operations specified by aiocb_list failed.
The application can check the status of each operation using
aio_return(3).

If lio_listio() fails with the error EAGAIN, EINTR, or EIO, then some
of the operations in aiocb_list may have been initiated. If lio_lis?
tio() fails for any other reason, then none of the I/O operations has
been initiated.

VERSIONS

The lio_listio() function is available since glibc 2.1.

ATTRIBUTES

For an explanation of the terms used in this section, see at?
tributes(7).

```
??????????????????????????????????????????
?Interface    ? Attribute     ? Value   ?
??????????????????????????????????????????
?lio_listio() ? Thread safety ? MT-Safe ?
??????????????????????????????????????????
```

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

It is a good idea to zero out the control blocks before use. The con?
trol blocks must not be changed while the I/O operations are in
progress. The buffer areas being read into or written from must not be
accessed during the operations or undefined results may occur. The
memory areas involved must remain valid.

Simultaneous I/O operations specifying the same aiocb structure produce undefined results.

SEE ALSO

aio_cancel(3), aio_error(3), aio_fsync(3), aio_return(3), aio_sus?
pend(3), aio_write(3), aio(7)

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/.