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 'lio_listio.3p' command

$ man lio_listio.3p

LIO_LISTIO(3P) POSIX Programmer's Manual LIO_LISTIO(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

lio_listio ? list directed I/O

SYNOPSIS

```
#include <aio.h>
int lio_listio(int mode, struct aiocb *restrict const list[restrict],
    int nent, struct sigevent *restrict sig);
```

DESCRIPTION

The lio_listio() function shall initiate a list of I/O requests with a single function call.

The mode argument takes one of the values LIO_WAIT or LIO_NOWAIT de? clared in <aio.h> and determines whether the function returns when the I/O operations have been completed, or as soon as the operations have been queued. If the mode argument is LIO_WAIT, the function shall wait until all I/O is complete and the sig argument shall be ignored.

If the mode argument is LIO_NOWAIT, the function shall return immedi? ately, and asynchronous notification shall occur, according to the sig argument, when all the I/O operations complete. If sig is NULL, then no

asynchronous notification shall occur. If sig is not NULL, asynchronous notification occurs as specified in Section 2.4.1, Signal Generation and Delivery when all the requests in list have completed.

The I/O requests enumerated by list are submitted in an unspecified or? der.

The list argument is an array of pointers to aiocb structures. The ar? ray contains nent elements. The array may contain NULL elements, which shall be ignored.

If the buffer pointed to by list or the aiocb structures pointed to by the elements of the array list become illegal addresses before all asynchronous I/O completed and, if necessary, the notification is sent, then the behavior is undefined. If the buffers pointed to by the aio_buf member of the aiocb structure pointed to by the elements of the array list become illegal addresses prior to the asynchronous I/O asso? ciated with that aiocb structure being completed, the behavior is unde? fined.

The aio_lio_opcode field of each aiocb structure specifies the opera? tion to be performed. The supported operations are LIO_READ, LIO_WRITE, and LIO_NOP; these symbols are defined in <aio.h>. The LIO_NOP opera? tion causes the list entry to be ignored. If the aio_lio_opcode element is equal to LIO_READ, then an I/O operation is submitted as if by a call to aio_read() with the aiocbp equal to the address of the aiocb structure. If the aio_lio_opcode element is equal to LIO_WRITE, then an I/O operation is submitted as if by a call to aio_write() with the aiocbp equal to the address of the aiocb structure.

The aio_fildes member specifies the file descriptor on which the opera? tion is to be performed.

The aio_buf member specifies the address of the buffer to or from which the data is transferred.

The aio_nbytes member specifies the number of bytes of data to be transferred.

The members of the aiocb structure further describe the I/O operation to be performed, in a manner identical to that of the corresponding

aiocb structure when used by the aio_read() and aio_write() functions.

The nent argument specifies how many elements are members of the list; that is, the length of the array.

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 associ? ated with aio_fildes.

For regular files, no data transfer shall occur past the offset maximum established in the open file description associated with aiocbp->aio_fildes.

If sig->sigev_notify is SIGEV_THREAD and sig->sigev_notify_attributes is a non-null pointer and the block pointed to by this pointer becomes an illegal address prior to all asynchronous I/O being completed, then the behavior is undefined.

RETURN VALUE

If the mode argument has the value LIO_NOWAIT, the lio_listio() func? tion shall return the value zero if the I/O operations are successfully queued; otherwise, the function shall return the value -1 and set errno to indicate the error.

If the mode argument has the value LIO_WAIT, the lio_listio() function shall return the value zero when all the indicated I/O has completed successfully. Otherwise, lio_listio() shall return a value of -1 and set errno to indicate the error.

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 shall examine the error status asso? ciated with each aiocb control block. The error statuses so returned are identical to those returned as the result of an aio_read() or aio_write() function.

ERRORS

The lio_listio() function shall fail if:

EAGAIN The resources necessary to queue all the I/O requests were not available. The application may check the error status for each aiocb to determine the individual request(s) that failed.

EAGAIN The number of entries indicated by nent would cause the system-wide limit {AIO_MAX} to be exceeded.

EINVAL The mode argument is not a proper value, or the value of nent was greater than {AIO_LISTIO_MAX}.

EINTR A signal was delivered while waiting for all I/O requests to complete during an LIO_WAIT operation. Note that, since each I/O operation invoked by lio_listio() may possibly provoke a signal when it completes, this error return may be caused by the com? pletion of one (or more) of the very I/O operations being awaited. Outstanding I/O requests are not canceled, and the ap? plication shall examine each list element to determine whether the request was initiated, canceled, or completed.

EIO One or more of the individual I/O operations failed. The appli? cation may check the error status for each aiocb structure to determine the individual request(s) that failed.

In addition to the errors returned by the lio_listio() function, if the lio_listio() function succeeds or fails with errors of [EAGAIN], [EINTR], or [EIO], then some of the I/O specified by the list may have been initiated. If the lio_listio() function fails with an error code other than [EAGAIN], [EINTR], or [EIO], no operations from the list shall have been initiated. The I/O operation indicated by each list el? ement can encounter errors specific to the individual read or write function being performed. In this event, the error status for each aiocb control block contains the associated error code. The error codes that can be set are the same as would be set by a read() or write() function, with the following additional error codes possible:

EAGAIN The requested I/O operation was not queued due to resource limi? tations.

ECANCELED

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

EFBIG The aiocbp->aio_lio_opcode is LIO_WRITE, the file is a regular file, aiocbp->aio_nbytes is greater than 0, and the aiocbp->aio_offset is greater than or equal to the offset maxi? mum in the open file description associated with aiocbp->aio_fildes.

EINPROGRESS

The requested I/O is in progress.

EOVERFLOW

The aiocbp->aio_lio_opcode is LIO_READ, the file is a regular file, aiocbp->aio_nbytes is greater than 0, and the aiocbp->aio_offset is before the end-of-file and is greater than or equal to the offset maximum in the open file description as? sociated with aiocbp->aio_fildes.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Although it may appear that there are inconsistencies in the specified circumstances for error codes, the [EIO] error condition applies when any circumstance relating to an individual operation makes that opera? tion fail. This might be due to a badly formulated request (for exam? ple, the aio_lio_opcode field is invalid, and aio_error() returns [EIN? VAL]) or might arise from application behavior (for example, the file descriptor is closed before the operation is initiated, and aio_error() returns [EBADF]).

The limitation on the set of error codes returned when operations from the list shall have been initiated enables applications to know when operations have been started and whether aio_error() is valid for a specific operation.

FUTURE DIRECTIONS

None.

SEE ALSO

aio_read(), aio_write(), aio_error(), aio_return(), aio_cancel(), close(), exec, exit(), fork(), lseek(), read()

The Base Definitions volume of POSIX.1?2017, <aio.h>