NAME

aioread, aiowrite – asynchronous I/O operations

SYNOPSIS

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

#include <sys/asynch.h>

int aioread(int *fildes*, **char** **bufp*, **int** *bufs*, **off_t** *offset*, **int** *whence*, **aio_result_t** **resultp***);**

int aiowrite(int *fildes*, **const char** **bufp*, **int** *bufs*, **off_t** *offset*, **int** *whence*, **aio_result_t** **resultp***);**

MT-LEVEL

Unsafe

DESCRIPTION

aioread( ) initiates one asynchronous **read**(2) and returns control to the calling program. The **read( )** continues concurrently with other activity of the process. An attempt is made to read *bufs* bytes of data from the object referenced by the descriptor *fildes* into the buffer pointed to by *bufp*.

aiowrite( ) initiates one asynchronous **write**(2) and returns control to the calling program. The **write( )** continues concurrently with other activity of the process. An attempt is made to write *bufs* bytes of data from the buffer pointed to by *bufp* to the object referenced by the descriptor *fildes*.

On objects capable of seeking, the I/O operation starts at the position specified by *whence* and *offset*. These parameters have the same meaning as the corresponding parameters to the **lseek**(2) function. On objects not capable of seeking the I/O operation always start from the current position and the parameters *whence* and *offset* are ignored. The seek pointer for objects capable of seeking is not updated by **aioread( )** or **aiowrite( )**. Sequential asynchronous operations on these devices must be managed by the application using the *whence* and *offset* parameters.

The result of the asynchronous operation is stored in the structure pointed to by *resultp*:

```
int aio_return;        /* return value of read( ) or write( ) */
int aio_errno;         /* value of errno for read( ) or write( ) */
```

Upon completion of the operation both *aio_return* and *aio_errno* are set to reflect the result of the operation. AIO_INPROGRESS is not a value used by the system so the client may detect a change in state by initializing *aio_return* to this value.

The application supplied buffer *bufp* should not be referenced by the application until after the operation has completed. While the operation is *in progress*, this buffer is in use by the operating system.

Notification of the completion of an asynchronous I/O operation may be obtained synchronously through the **aiowait**(3) function, or asynchronously by installing a signal handler for the **SIGIO** signal. Asynchronous notification is accomplished by sending the process a **SIGIO** signal. If a signal handler is not installed for the **SIGIO** signal, asynchronous notification is disabled. The delivery of this instance of the **SIGIO** signal is reliable in that a signal delivered while the handler is executing is not lost. If the client ensures that **aiowait**(3) returns nothing (using a polling timeout) before returning from the signal handler, no asynchronous I/O notifications are lost. The **aiowait**(3) function is the only way to dequeue an asynchronous notification. Note: **SIGIO** may have several meanings simultaneously: for example, that a descriptor generated **SIGIO** and an asynchronous operation completed. Further, issuing an asynchronous request successfully guarantees that space exists to queue the completion notification.

close(2), **exit**(2) and **execve( )** (see **exec**(2)) will block until all pending asynchronous I/O operations can be canceled by the system.

It is an error to use the same result buffer in more than one outstanding request. These structures may only be reused after the system has completed the operation.

RETURN VALUES

aioread( ) and **aiowrite( )** return:

0 on success.

–1 on failure and set **errno** to indicate the error.

ERRORS

EAGAIN The number of asynchronous requests that the system can handle at any one time has been exceeded

EBADF *fildes* is not a valid file descriptor open for reading.

EFAULT At least one of *bufp* points to an address outside the address space of the requesting process. See **NOTES** below.

EINVAL The parameter *resultp* is currently being used by an outstanding asynchronous request.

ENOMEM Memory resources are unavailable to initiate request.

SEE ALSO

close(2), **exec**(2), **exit**(2), **lseek**(2), **open**(2), **read**(2), **write**(2), **aiocancel**(3), **aiowait**(3), **sigvec**(3B)

NOTES

Passing an illegal address to *bufp* will result in setting **errno** to EFAULT *only* if it is detected by the application process.