NAME

aio_read, aio_write – asynchronous read and write operations

SYNOPSIS

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

```
#include <aio.h>

int aio_read(struct aiocb *aiocbp);

int aio_write(struct aiocb *aiocbp);

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

aio_read( ) queues an asynchronous read request, and returns control immediately. Rather than blocking until completion, the read operation continues concurrently with other activity of the process.

Upon enqueuing the request, the calling process reads *aiocbp*-**>nbytes** from the file referred to by *aiocbp*-**>fildes** into the buffer pointed to by *aiocbp*-**>aio_buf**. *aiocbp*-**>offset** marks the absolute position from the beginning of the file (in bytes) at which the read begins.

aio_write( ) queues an asynchronous write request, and returns control immediately. Rather than blocking until completion, the write operation continues concurrently with other activity of the process.

Upon enqueuing the request, the calling process writes *aiocbp*-**>nbytes** from the buffer pointed to by *aiocbp*-**>aio_buf** into the file referred to by *aiocbp*-**>fildes**. If **O_APPEND** is set for *aiocbp*-**>fildes**, **aio_write( )** operations append to the file in the same order as the calls were made.

If **O_APPEND** is not set for the file descriptor, then the write operation will occur at the absolute position from the beginning of the file plus *aiocbp*->**offset** (in bytes).

These asynchronous operations are submitted at a priority equal to the calling process' scheduling priority minus *aiocbp*->**aio_reqprio**.

aiocb->**aio_sigevent** defines both the signal to be generated and how the calling process will be notified upon I/O completion. If **aio_sigevent.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 **aio_sigevent.sigev_notify** is **SIGEV_SIGNAL**, then the signal specified in **aio_sigevent.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 **aio_sigevent.sigev_value** will be the **si_value** component of the generated signal (see **siginfo**(5)).

RETURN VALUES

If the I/O operation is successfully queued, **aio_read()** and **aio_write()** return **0,** otherwise, they return **-1,** and set **errno** to indicate the error condition. *aiocbp* may be used as an argument to **aio_error**(3R) and **aio_return**(3R) in order to determine the error status and the return status of the asynchronous operation while it is proceeding.

ERRORS

EAGAIN	The requested asynchronous I/O operation was not queued due to system resource limitations.
ENOSYS	**aio_read()** or is not supported by this implementation.
EBADF	If the calling function is **aio_read(),** and *aiocbp*->**fildes** is not a valid file descriptor open for reading. If the calling function is **aio_write(),** and *aiocbp*->**fildes** is not a valid file descriptor open for writing.
EINVAL	• The file offset value implied by *aiocbp*->**aio_offset** would be invalid, • *aiocbp*->**aio_reqprio** is not a valid value, or • *aiocbp*->**aio_nbytes** is an invalid value.
ECANCELED	The requested I/O was canceled before the I/O completed due to an explicit **aio_cancel**(3R) request.
EINVAL	The file offset value implied by *aiocbp*->**aio_offset** would be invalid.

SEE ALSO

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

NOTES

For portability, the application should set *aiocb*->**aio_reqprio** to **0.**

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.