NAME

aio_return, aio_error – retrieve return or error status of asynchronous I/O operation

SYNOPSIS

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

```
#include <aio.h>

ssize_t aio_return(struct aiocb * aiocbp);

int aio_error(const 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

Async-Signal-Safe

DESCRIPTION

aio_return( ) returns the return status of the asynchronous I/O request associated with the **aiocb** structure pointed to by *aiocbp.*

aio_error( ) returns the error status of the asynchronous I/O request associated with the **aiocb** structure pointed to by *aiocbp.*

aio_return( ) should be called only once to retrieve the valid return status of a given asynchronous operation, after **aio_error( )** has returned a value other than **EINPROGRESS.**

RETURN VALUES

If the asynchronous I/O operation has completed successfully, **aio_return( )** returns the return status, as described for **read**(2), **write**(2), and **fsync**(3C).

If the asynchronous I/O operation has completed successfully, **aio_error( )** returns **0.** If the operation has not yet completed, then **EINPROGRESS** is returned. If the asynchronous I/O operation has completed unsuccessfully, then the error status, as described for **read**(2), **write**(2), and **fsync**(3C) is returned.

If unsuccessful, **aio_return( )** or **aio_error( )** return **-1,** and set **errno** to indicate the error condition.

ERRORS

EINVAL *aiocbp* does not reference an asynchronous operation which has completed or failed.

ENOSYS **aio_return( )** or **aio_error( )** is not supported by this implementation.

EXAMPLES

```
#include <aio.h>
#include <errno.h>
#include <signal.h>
struct aiocb        my_aiocb;
struct sigaction    my_sigaction;
void                my_aio_handler(int, siginfo_t *, void *);
...
my_sigaction.sa_flags = SA_SIGINFO;
my_sigaction.sa_sigaction = my_aio_handler;
sigsetempty(&my_sigaction.sa_mask);
(void) sigaction(SIGRTMIN, &my_sigaction, NULL);
...
my_aiocb.aio_sigevent.sigev_notify = SIGEV_SIGNAL;
my_aiocb.aio_sigevent.sigev_signo = SIGRTMIN;
my_aiocb.aio_sigevent.sigev_value.sival_ptr = &myaiocb;
...
(void) aio_read(&my_aiocb);
...
void
my_aio_handler(int signo, siginfo_t *siginfo, void *context) {
int    my_errno;
struct aiocb  *my_aiocbp;
my_aiocbp = siginfo.si_value.sival_ptr;
      if ((my_errno = aio_error(my_aiocb)) != EINPROGRESS) {
            int       my_status = aio_return(my_aiocb);
            if (my_status >= 0){/* start another operation */
                  ...
            } else    {                    /* handle I/O error */
                  ...
            }
      }
}
```

SEE ALSO

close(2), **exec**(2), **exit**(2), **fork**(2), **lseek**(2), **read**(2), **write**(2), **fsync**(3C), **aio_cancel**(3R), **aio_fsync**(3R), **aio_read**(3R), **lio_listio**(3R)

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.