NAME

aio_cancel – cancel asynchronous I/O request

SYNOPSIS

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

```
#include <aio.h>
int aio_cancel(int fildes, struct aiocb *aiocbp );
```

MT-LEVEL

MT-Safe

DESCRIPTION

aio_cancel( ) attempts to cancel either one or all outstanding asynchronous I/O requests pending on the file descriptor specified by *fildes.* If *aiocbp* is **NULL**, then all such outstanding cancelable requests are canceled; otherwise, the individual request referenced by *aiocbp* references will be canceled.

Normal completion notification occurs even for asynchronous I/O operations that are successfully canceled. If there are requests which cannot be canceled, then the normal asynchronous completion process takes place for those requests, and their associated **aiocb** structures are not modified.

```
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 */
};
```

RETURN VALUES

If the requested operation(s) were canceled, **aio_cancel( )** returns **AIO_CANCELED**. But if at least one of the requested operation(s) cannot be canceled because it is in progress, then **AIO_NOTCANCELED** is returned, and the application may determine the state of affairs for these operation(s) by using **aio_error**(3R). If all of the operation(s) had already completed, **AIO_ALLDONE** is returned. Otherwise, **aio_cancel( )** returns **-1,** and sets **errno** to indicate the error condition.

ERRORS

EBADF *fildes* is not a valid file descriptor.

ENOSYS **aio_cancel( )** is not supported by this implementation.

SEE ALSO

aio_return(3R), **aio_read**(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.