NAME

aio_fsync – asynchronous file synchronization

SYNOPSIS

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

#include <aio.h>

int aio_fsync(int *op***, aiocb** **aiocbp***);**

MT-LEVEL

MT-Safe

DESCRIPTION

aio_fsync( ) queues an asynchronous **fsync**(3C) or **fdatasync**(3R) request for all the currently queued I/O operations on the file referenced by *aiocbp*->**aio_fildes,** and returns control immediately. This request is serviced concurrently with other activity of the process. If *op* is **O_DSYNC**, all I/O operations are completed by a call to **fdatasync**(3R) (synchronized I/O data integrity completion). If *op* is **O_SYNC**, all I/O operations are completed by a call to **fsync**(3C) (synchronized I/O file integrity completion). (see **fcntl**(5) definitions of **O_DSYNC** and **O_SYNC**.)

When the request is queued, the error status for the operation is **EINPROGRESS**. When all data has been successfully transferred, the error status is reset to reflect the success or failure of the operation. **aio_return**(3R) and **aio_error**(3R) may be used with this *aiocbp* value to monitor both the return and the error status of the asynchronous operation while it is proceeding.

aiocbp->**aio_sigevent** defines the signal to be generated upon I/O completion.
If *aiocbp*->**aio_sigevent.sigev_signo** is non-zero, then a signal will be generated when all I/O operations have achieved syncronized I/O completion.

```
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 I/O operation is successfully queued, **aio_fsync( )** returns **0.** Otherwise, it returns **-1,** and sets **errno** to indicate the error condition.

ERRORS

EAGAIN	The requested asynchronous operation was not queued due to temporary resource limitations.
EBADF	*aiocbp*->**aio_fildes** is not a valid file descriptor open for writing.
EINVAL	This implementation does not support synchronized I/O for this file.
	A value of *op* other than **O_DSYNC** or **O_SYNC** was specified.
ENOSYS	**aio_fsync( )** is not supported by this implementation.

SEE ALSO

fcntl(2), **open**(2), **read**(2), **write**(2), **fsync**(3C), **aio_error**(3R), **aio_return**(3R), **fdatasync**(3R), **fcntl**(5)

NOTES

If **aio_fsync( )** fails, outstanding I/O operations are not guaranteed to have been completed.

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.