NAME

aio_suspend – wait for asynchronous I/O request

SYNOPSIS

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

#include <aio.h>

int aio_suspend(const struct aiocb ∗ **const** *list*[ ]**, int** *nent*,
    const struct timespec ∗*timeout***);**

MT-LEVEL

Async-Signal-Safe

DESCRIPTION

aio_suspend( ) suspends the caller until at least one of the asynchronous I/O operations referenced by *list* has completed, until a signal interrupts the function, or, if *timeout* is not NULL, until the time interval specified by *timeout* has passed. If any of the **aiocb** structures in the list corresponds to a completed asynchronous I/O operation (i.e., the error status for the operation is not equal to **EINPROGRESS**), at the time of the call, the function returns without suspending the caller.

If the time interval indicated in the **timespec** structure pointed to by *timeout* passes before any of the I/O operations referenced by *list* are completed, then **aio_suspend( )** returns with an error.

list is an array of pointers to asynchronous I/O control blocks. *nent* indicates the number of elements in this array. Each **aiocb** structure pointed to must have been used in initiating an asynchronous I/O request via **aio_read**(3R), **aio_write**(3R), **aio_fsync**(3R), or **lio_listio**(3R). This array may contain NULL pointers which will be ignored.

```
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 */
};
```

```
struct timespec {
  time_t        tv_sec;        /* seconds */
  long          tv_nsec;       /* and nanoseconds */
};
```

RETURN VALUES

If **aio_suspend( )** returns after one or more asynchronous I/O operations have completed, it returns **0.** Otherwise, it returns **-1,** and sets **errno** to indicate the error condition.

The application may determine which asynchronous I/O had completed with both the associated error and return status of **aio_return**(3R), and **aio_error**(3R).

ERRORS

EAGAIN — No asynchronous I/O indicated in the list referenced by *list* completed in the time interval indicated by *timeout.*

EINTR — A signal interrupted the **aio_suspen( )** function. Note that, since each asynchronous I/O operation may possibly provoke a signal when it completes, this error return may be caused by the completion of one (or more) of the very I/O operations being awaited.

ENOSYS — **aio_suspend( )** is not supported by this implementation.

SEE ALSO

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