

**NAME**

`aio_read` - asynchronous read

**SYNOPSIS**

```
#include <aio.h>
```

```
int aio_read(struct aiocb *aiocbp);
```

Link with `-lrt`.

**DESCRIPTION**

The `aio_read()` function queues the I/O request described by the buffer pointed to by `aiocbp`. This function is the asynchronous analog of `read(2)`. The arguments of the call

```
read(fd, buf, count)
```

correspond (in order) to the fields `aio_fildes`, `aio_buf`, and `aio_nbytes` of the structure pointed to by `aiocbp`. (See `aio(7)` for a description of the `aiocb` structure.)

The data is read starting at the absolute file offset `aiocbp->aio_offset`, regardless of the current file offset. After the call, the value of the current file offset is unspecified.

The "asynchronous" means that this call returns as soon as the request has been enqueued; the read may or may not have completed when the call returns. One tests for completion using `aio_error(3)`. The return status of a completed I/O operation can be obtained by `aio_return(3)`. Asynchronous notification of I/O completion can be obtained by setting `aiocbp->aio_sigevent` appropriately; see `sigevent(7)` for details.

If `_POSIX_PRIORITYIZED_IO` is defined, and this file supports it, then the asynchronous operation is submitted at a priority equal to that of the calling process minus `aiocbp->aio_reqprio`.

The field `aiocbp->aio_lio_opcode` is ignored.

No data is read from a regular file beyond its maximum offset.

**RETURN VALUE**

On success, 0 is returned. On error, the request is not enqueued, -1 is returned, and `errno` is set appropriately. If an error is detected only later, it will be reported via `aio_return(3)` (returns status -1) and `aio_error(3)` (error status—whatever one would have gotten in `errno`, such as **EBADF**).

**ERRORS****EAGAIN**

Out of resources.

**EBADF**

`aio_fildes` is not a valid file descriptor open for reading.

**EINVAL**

One or more of `aio_offset`, `aio_reqprio`, or `aio_nbytes` are invalid.

**ENOSYS**

`aio_read()` is not implemented.

**E\_OVERFLOW**

The file is a regular file, we start reading before end-of-file and want at least one byte, but the starting position is past the maximum offset for this file.

**VERSIONS**

The `aio_read()` function is available since glibc 2.1.

**CONFORMING TO**

POSIX.1-2001, POSIX.1-2008.

**NOTES**

It is a good idea to zero out the control block before use. The control block must not be changed while the read operation is in progress. The buffer area being read into must not be accessed during the operation or undefined results may occur. The memory areas involved must remain valid.

Simultaneous I/O operations specifying the same *aiocb* structure produce undefined results.

**EXAMPLE**

See [aio\(7\)](#).

**SEE ALSO**

[aio\\_cancel\(3\)](#), [aio\\_error\(3\)](#), [aio\\_fsync\(3\)](#), [aio\\_return\(3\)](#), [aio\\_suspend\(3\)](#), [aio\\_write\(3\)](#), [lio\\_listio\(3\)](#), [aio\(7\)](#)

**COLOPHON**

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <http://www.kernel.org/doc/man-pages/>.