NAME

io_getevents – read asynchronous I/O events from the completion queue

SYNOPSIS

#include <linux/aio_abi.h> /* Defines needed types */
#include <linux/time.h> /* Defines 'struct timespec' */

int io_getevents(aio_context_t *ctx_id***, long** *min_nr***, long** *nr***,**
struct io_event ****events*, struct timespec ****timeout***);**

Note: There is no glibc wrapper for this system call; see NOTES.

DESCRIPTION

The **io_getevents**() system call attempts to read at least *min_nr* events and up to *nr* events from the completion queue of the AIO context specified by *ctx_id*.

The *timeout* argument specifies the amount of time to wait for events, and is specified as a relative timeout in a structure of the following form:

```
struct timespec {
    time_t tv_sec;      /* seconds */
    long   tv_nsec;     /* nanoseconds [0 .. 999999999] */
};
```

The specified time will be rounded up to the system clock granularity and is guaranteed not to expire early.

Specifying *timeout* as NULL means block indefinitely until at least *min_nr* events have been obtained.

RETURN VALUE

On success, **io_getevents**() returns the number of events read. This may be 0, or a value less than *min_nr*, if the *timeout* expired. It may also be a nonzero value less than *min_nr*, if the call was interrupted by a signal handler.

For the failure return, see NOTES.

ERRORS

EFAULT
Either *events* or *timeout* is an invalid pointer.

EINTR
Interrupted by a signal handler; see **signal**(7).

EINVAL
ctx_id is invalid. *min_nr* is out of range or *nr* is out of range.

ENOSYS
io_getevents() is not implemented on this architecture.

VERSIONS

The asynchronous I/O system calls first appeared in Linux 2.5.

CONFORMING TO

io_getevents() is Linux-specific and should not be used in programs that are intended to be portable.

NOTES

Glibc does not provide a wrapper function for this system call. You could invoke it using **syscall**(2). But instead, you probably want to use the **io_getevents**() wrapper function provided by *libaio*.

Note that the *libaio* wrapper function uses a different type (*io_context_t*) for the *ctx_id* argument. Note also that the *libaio* wrapper does not follow the usual C library conventions for indicating errors: on error it returns a negated error number (the negative of one of the values listed in ERRORS). If the system call is invoked via **syscall**(2), then the return value follows the usual conventions for indicating an error: −1, with *errno* set to a (positive) value that indicates the error.

BUGS

An invalid *ctx_id* may cause a segmentation fault instead of generating the error **EINVAL**.

SEE ALSO

io_cancel(2), **io_destroy**(2), **io_setup**(2), **io_submit**(2), **aio**(7), **time**(7)

COLOPHON

This page is part of release 5.05 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 https://www.kernel.org/doc/man−pages/.