

**NAME**

`io_submit` – submit asynchronous I/O blocks for processing

**SYNOPSIS**

```
#include <linux/aio_abi.h>      /* Defines needed types */
```

```
int io_submit(aio_context_t ctx_id, long nr, struct iocb **iocbpp);
```

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

**DESCRIPTION**

The `io_submit()` system call queues *nr* I/O request blocks for processing in the AIO context *ctx\_id*. The *iocbpp* argument should be an array of *nr* AIO control blocks, which will be submitted to context *ctx\_id*.

The *iocb* (I/O control block) structure defined in *linux/aio\_abi.h* defines the parameters that control the I/O operation.

```
#include <linux/aio_abi.h>

struct iocb {
    __u64    aio_data;
    __u32    PADDED(aio_key, aio_rw_flags);
    __u16    aio_lio_opcode;
    __s16    aio_reqprio;
    __u32    aio_fildes;
    __u64    aio_buf;
    __u64    aio_nbytes;
    __s64    aio_offset;
    __u64    aio_reserved2;
    __u32    aio_flags;
    __u32    aio_resfd;
};
```

The fields of this structure are as follows:

*aio\_data*

This data is copied into the *data* field of the *io\_event* structure upon I/O completion (see `io_getevents(2)`).

*aio\_key*

This is an internal field used by the kernel. Do not modify this field after an `io_submit(2)` call.

*aio\_rw\_flags*

This defines the R/W flags passed with structure. The valid values are:

**RWF\_APPEND** (since Linux 4.16)

Append data to the end of the file. See the description of the flag of the same name in `pwritev2(2)` as well as the description of **O\_APPEND** in `open(2)`. The *aio\_offset* field is ignored. The file offset is not changed.

**RWF\_DSYNC** (since Linux 4.7)

Write operation complete according to requirement of synchronized I/O data integrity. See the description of the flag of the same name in `pwritev2(2)` as well the description of **O\_DSYNC** in `open(2)`.

**RWF\_HIPRI** (since Linux 4.6)

High priority request, poll if possible

**RWF\_NOWAIT** (since Linux 4.14)

Don't wait if the I/O will block for operations such as file block allocations, dirty page flush, mutex locks, or a congested block device inside the kernel. If any of these conditions are met, the control block is returned immediately with a return value of **-EAGAIN** in the *res* field of the *io\_event* structure (see `io_getevents(2)`).

**RWF\_SYNC** (since Linux 4.7)

Write operation complete according to requirement of synchronized I/O file integrity. See the description of the flag of the same name in **pwritev2(2)** as well the description of **O\_SYNC** in **open(2)**.

*aio\_lio\_opcode*

This defines the type of I/O to be performed by the *iocb* structure. The valid values are defined by the enum defined in *linux/aio\_abi.h*:

```
enum {
    IOCB_CMD_PREAD = 0,
    IOCB_CMD_PWRITE = 1,
    IOCB_CMD_FSYNC = 2,
    IOCB_CMD_FDSYNC = 3,
    IOCB_CMD_NOOP = 6,
    IOCB_CMD_PREADV = 7,
    IOCB_CMD_PWRITEV = 8,
};
```

*aio\_reqprio*

This defines the requests priority.

*aio\_fildes*

The file descriptor on which the I/O operation is to be performed.

*aio\_buf*

This is the buffer used to transfer data for a read or write operation.

*aio\_nbytes*

This is the size of the buffer pointed to by *aio\_buf*.

*aio\_offset*

This is the file offset at which the I/O operation is to be performed.

*aio\_flags*

This is the set of flags associated with the *iocb* structure. The valid values are:

**IOCB\_FLAG\_RESFD**

Asynchronous I/O control must signal the file descriptor mentioned in *aio\_resfd* upon completion.

**IOCB\_FLAG\_IOPRIO** (since Linux 4.18)

Interpret the *aio\_reqprio* field as an **IOPRIO\_VALUE** as defined by *linux/ioprio.h*.

*aio\_resfd*

The file descriptor to signal in the event of asynchronous I/O completion.

**RETURN VALUE**

On success, **io\_submit()** returns the number of *iocbs* submitted (which may be less than *nr*, or 0 if *nr* is zero). For the failure return, see NOTES.

**ERRORS****EAGAIN**

Insufficient resources are available to queue any *iocbs*.

**EBADF**

The file descriptor specified in the first *iocb* is invalid.

**EFAULT**

One of the data structures points to invalid data.

**EINVAL**

The AIO context specified by *ctx\_id* is invalid. *nr* is less than 0. The *iocb* at *\*iocbpp[0]* is not properly initialized, the operation specified is invalid for the file descriptor in the *iocb*, or the value

in the *aio\_reqprio* field is invalid.

**ENOSYS**

**io\_submit()** is not implemented on this architecture.

**EPERM**

The *aio\_reqprio* field is set with the class **IOPRIO\_CLASS\_RT**, but the submitting context does not have the **CAP\_SYS\_ADMIN** capability.

**VERSIONS**

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

**CONFORMING TO**

**io\_submit()** 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\_submit()** 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.

**SEE ALSO**

**io\_cancel(2)**, **io\_destroy(2)**, **io\_getevents(2)**, **io\_setup(2)**, **aio(7)**

**COLOPHON**

This page is part of release 5.02 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/>.