NAME

io_destroy − destroy an asynchronous I/O context

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <linux/aio_abi.h>    /* Definition of aio_context_t */
#include <sys/syscall.h>      /* Definition of SYS_* constants */
#include <unistd.h>

int syscall(SYS_io_destroy, aio_context_t ctx_id);
```

Note: glibc provides no wrapper for **io_destroy**(), necessitating the use of **syscall**(2).

DESCRIPTION

Note: this page describes the raw Linux system call interface. The wrapper function provided by *libaio* uses a different type for the *ctx_id* argument. See NOTES.

The **io_destroy**() system call will attempt to cancel all outstanding asynchronous I/O operations against *ctx_id*, will block on the completion of all operations that could not be canceled, and will destroy the *ctx_id*.

RETURN VALUE

On success, **io_destroy**() returns 0. For the failure return, see NOTES.

ERRORS

EFAULT
: The context pointed to is invalid.

EINVAL
: The AIO context specified by *ctx_id* is invalid.

ENOSYS
: **io_destroy**() is not implemented on this architecture.

VERSIONS

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

STANDARDS

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

NOTES

You probably want to use the **io_destroy**() 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_getevents**(2), **io_setup**(2), **io_submit**(2), **aio**(7)