
*Full credit is given to the above companies including the OS that this PDF file was generated!*

## *Red Hat Enterprise Linux Release 9.2 Manual Pages on 'io_setup.2' command*

***$ man io_setup.2***

IO_SETUP(2) Linux Programmer's Manual IO_SETUP(2)

NAME

io_setup - create an asynchronous I/O context

SYNOPSIS

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

long io_setup(unsigned nr_events, aio_context_t *ctx_idp);

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

DESCRIPTION

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

The io_setup() system call creates an asynchronous I/O context suitable for concurrently processing nr_events operations. The ctx_idp argument must not point to an AIO context that already exists, and must be ini? tialized to 0 prior to the call. On successful creation of the AIO context, *ctx_idp is filled in with the resulting handle.

RETURN VALUE

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

ERRORS

EAGAIN The specified nr_events exceeds the limit of available events, as defined in /proc/sys/fs/aio-max-nr (see proc(5)).

EFAULT An invalid pointer is passed for ctx_idp.

EINVAL ctx_idp is not initialized, or the specified nr_events exceeds

internal limits. nr_events should be greater than 0.

ENOMEM Insufficient kernel resources are available.

ENOSYS io_setup() is not implemented on this architecture.

## VERSIONS

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

## CONFORMING TO

io_setup() 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_setup() wrapper function provided by libaio.

Note that the libaio wrapper function uses a different type (io_con? text_t *) for the ctx_idp 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 in? dicating an error: -1, with errno set to a (positive) value that indi? cates the error.

## SEE ALSO

io_cancel(2), io_destroy(2), io_getevents(2), io_submit(2), aio(7)

## COLOPHON

This page is part of release 5.10 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/.