NAME

fsync, fdatasync − synchronize a file's in-core state with storage device

SYNOPSIS

#include <unistd.h>

int fsync(int *fd***);**

int fdatasync(int *fd***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

fsync(): _BSD_SOURCE || _XOPEN_SOURCE
|| /* since glibc 2.8: */ _POSIX_C_SOURCE >= 200112L
fdatasync(): _POSIX_C_SOURCE >= 199309L || _XOPEN_SOURCE >= 500

DESCRIPTION

fsync() transfers ("flushes") all modified in-core data of (i.e., modified buffer cache pages for) the file referred to by the file descriptor *fd* to the disk device (or other permanent storage device) where that file resides. The call blocks until the device reports that the transfer has completed. It also flushes metadata information associated with the file (see **stat**(2)).

Calling **fsync**() does not necessarily ensure that the entry in the directory containing the file has also reached disk. For that an explicit **fsync**() on a file descriptor for the directory is also needed.

fdatasync() is similar to **fsync**(), but does not flush modified metadata unless that metadata is needed in order to allow a subsequent data retrieval to be correctly handled. For example, changes to *st_atime* or *st_mtime* (respectively, time of last access and time of last modification; see **stat**(2)) do not require flushing because they are not necessary for a subsequent data read to be handled correctly. On the other hand, a change to the file size (*st_size*, as made by say **ftruncate**(2)), would require a metadata flush.

The aim of **fdatasync**() is to reduce disk activity for applications that do not require all metadata to be synchronized with the disk.

RETURN VALUE

On success, these system calls return zero. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EBADF
fd is not a valid file descriptor open for writing.

EIO An error occurred during synchronization.

EROFS, **EINVAL**
fd is bound to a special file which does not support synchronization.

CONFORMING TO

4.3BSD, POSIX.1-2001.

AVAILABILITY

On POSIX systems on which **fdatasync**() is available, **_POSIX_SYNCHRONIZED_IO** is defined in *<unistd.h>* to a value greater than 0. (See also **sysconf**(3).)

NOTES

Applications that access databases or log files often write a tiny data fragment (e.g., one line in a log file) and then call **fsync**() immediately in order to ensure that the written data is physically stored on the hard-disk. Unfortunately, **fsync**() will always initiate two write operations: one for the newly written data and another one in order to update the modification time stored in the inode. If the modification time is not a part of the transaction concept **fdatasync**() can be used to avoid unnecessary inode disk write operations.

If the underlying hard disk has write caching enabled, then the data may not really be on permanent storage when **fsync**() / **fdatasync**() return.

When an ext2 file system is mounted with the *sync* option, directory entries are also implicitly synced by **fsync**().

On kernels before 2.4, **fsync**() on big files can be inefficient. An alternative might be to use the **O_SYNC** flag to **open**(2).

In Linux 2.2 and earlier, **fdatasync**() is equivalent to **fsync**(), and so has no performance advantage.

SEE ALSO

bdflush(2), **open**(2), **sync**(2), **sync_file_range**(2), **hdparm**(8), **mount**(8), **sync**(8), **update**(8)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.