NAME

copy_file_range - kernel copy of a byte range from one file to another or within one file

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/types.h>
#include <unistd.h>

ssize_t
copy_file_range(*int infd, off_t *inoffp, int outfd, off_t *outoffp, size_t len, unsigned int flags*);

DESCRIPTION

The **copy_file_range**() system call copies up to *len* bytes from *infd* to *outfd* in the kernel. It may do this using a file system specific technique if *infd* and *outfd* are on the same file system. If *infd* and *outfd* refer to the same file, the byte ranges defined by the input file offset, output file offset and *len* cannot overlap. The *infd* argument must be opened for reading and the *outfd* argument must be opened for writing, but not O_APPEND. If *inoffp* or *outoffp* is NULL, the file offset for *infd* or *outfd* respectively will be used and updated by the number of bytes copied. If *inoffp* or *outoffp* is not NULL, the byte offset pointed to by *inoffp* or *outoffp* respectively will be used/updated and the file offset for *infd* or *outfd* respectively will not be affected. The *flags* argument must be 0.

This system call attempts to maintain holes in the output file for the byte range being copied. However, this does not always work well. It is recommended that sparse files be copied in a loop using lseek(2) with SEEK_HOLE, SEEK_DATA arguments and this system call for the data ranges found.

For best performance, call **copy_file_range**() with the largest *len* value possible. It is interruptible on most file systems, so there is no penalty for using very large len values, even SSIZE_MAX.

RETURN VALUES

If it succeeds, the call returns the number of bytes copied, which can be fewer than *len*. Returning fewer bytes than *len* does not necessarily indicate that EOF was reached. However, a return of zero for a non-zero *len* argument indicates that the offset for *infd* is at or beyond EOF. **copy_file_range**() should be used in a loop until copying of the desired byte range has been completed. If an error has occurred, a -1 is returned and the error code is placed in the global variable *errno*.

ERRORS

The **copy_file_range**() system call will fail if:

[EBADF] If *infd* is not open for reading or *outfd* is not open for writing, or opened for writing with O_APPEND, or if *infd* and *outfd* refer to the same file.

[EFBIG] If the copy exceeds the process's file size limit or the maximum file size for the file system *outfd* resides on.

[EINTR] A signal interrupted the system call before it could be completed. This may happen for files on some NFS mounts. When this happens, the values pointed to by *inoffp* and *outoffp* are reset to the initial values for the system call.

[EINVAL] *infd* and *outfd* refer to the same file and the byte ranges overlap or *flags* is not zero.

[EIO] An I/O error occurred while reading/writing the files.

[EINTEGRITY] Corrupted data was detected while reading from a file system.

[EISDIR] If either *infd* or *outfd* refers to a directory.

[ENOSPC] File system that stores *outfd* is full.

SEE ALSO

lseek(2)

STANDARDS

The **copy_file_range**() system call is expected to be compatible with the Linux system call of the same name.

HISTORY

The **copy_file_range**() function appeared in FreeBSD 13.0.