NAME

link − make a new name for a file

SYNOPSIS

#include <unistd.h>

int link(const char **oldpath*, const char **newpath*);

DESCRIPTION

link() creates a new link (also known as a hard link) to an existing file.

If *newpath* exists it will *not* be overwritten.

This new name may be used exactly as the old one for any operation; both names refer to the same file (and so have the same permissions and ownership) and it is impossible to tell which name was the "original".

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EACCES
Write access to the directory containing *newpath* is denied, or search permission is denied for one of the directories in the path prefix of *oldpath* or *newpath*. (See also **path_resolution**(7).)

EEXIST
newpath already exists.

EFAULT
oldpath or *newpath* points outside your accessible address space.

EIO An I/O error occurred.

ELOOP
Too many symbolic links were encountered in resolving *oldpath* or *newpath*.

EMLINK
The file referred to by *oldpath* already has the maximum number of links to it.

ENAMETOOLONG
oldpath or *newpath* was too long.

ENOENT
A directory component in *oldpath* or *newpath* does not exist or is a dangling symbolic link.

ENOMEM
Insufficient kernel memory was available.

ENOSPC
The device containing the file has no room for the new directory entry.

ENOTDIR
A component used as a directory in *oldpath* or *newpath* is not, in fact, a directory.

EPERM
oldpath is a directory.

EPERM
The file system containing *oldpath* and *newpath* does not support the creation of hard links.

EROFS
The file is on a read-only file system.

EXDEV
oldpath and *newpath* are not on the same mounted file system. (Linux permits a file system to be mounted at multiple points, but **link**() does not work across different mount points, even if the

same file system is mounted on both.)

CONFORMING TO

SVr4, 4.3BSD, POSIX.1-2001 (but see NOTES).

NOTES

Hard links, as created by **link**(), cannot span file systems. Use **symlink**(2) if this is required.

POSIX.1-2001 says that **link**() should dereference *oldpath* if it is a symbolic link. However, since kernel 2.0, Linux does not do so: if *oldpath* is a symbolic link, then *newpath* is created as a (hard) link to the same symbolic link file (i.e., *newpath* becomes a symbolic link to the same file that *oldpath* refers to). Some other implementations behave in the same manner as Linux. POSIX.1-2008 changes the specification of **link**(), making it implementation-dependent whether or not *oldpath* is dereferenced if it is a symbolic link. For precise control over the treatment of symbolic links when creating a link, see **linkat**(2).

BUGS

On NFS file systems, the return code may be wrong in case the NFS server performs the link creation and dies before it can say so. Use **stat**(2) to find out if the link got created.

SEE ALSO

ln(1), **linkat**(2), **open**(2), **rename**(2), **stat**(2), **symlink**(2), **unlink**(2), **path_resolution**(7), **symlink**(7)

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/.