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 'symlink.2' command

$ man symlink.2

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

NAME

symlink, symlinkat - make a new name for a file

SYNOPSIS

#include <unistd.h>

int symlink(const char *target, const char *linkpath);

#include <fcntl.h> /* Definition of AT_* constants */

#include <unistd.h>

int symlinkat(const char *target, int newdirfd, const char *linkpath);

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

symlink():

_XOPEN_SOURCE >= 500 || _POSIX_C_SOURCE >= 200112L

|| /* Glibc versions <= 2.19: */ _BSD_SOURCE

symlinkat():

Since glibc 2.10:

_POSIX_C_SOURCE >= 200809L

Before glibc 2.10:

_ATFILE_SOURCE

DESCRIPTION

symlink() creates a symbolic link named linkpath which contains the

string target.

Symbolic links are interpreted at run time as if the contents of the

link had been substituted into the path being followed to find a file

or directory.

Symbolic links may contain .. path components, which (if used at the start of the link) refer to the parent directories of that in which the link resides.

A symbolic link (also known as a soft link) may point to an existing file or to a nonexistent one; the latter case is known as a dangling link.

The permissions of a symbolic link are irrelevant; the ownership is ig? nored when following the link, but is checked when removal or renaming of the link is requested and the link is in a directory with the sticky bit (S_ISVTX) set.

If linkpath exists, it will not be overwritten.

symlinkat()

The symlinkat() system call operates in exactly the same way as sym? link(), except for the differences described here.

If the pathname given in linkpath is relative, then it is interpreted relative to the directory referred to by the file descriptor newdirfd (rather than relative to the current working directory of the calling process, as is done by symlink() for a relative pathname).

If linkpath is relative and newdirfd is the special value AT_FDCWD, then linkpath is interpreted relative to the current working directory of the calling process (like symlink()).

If linkpath is absolute, then newdirfd is ignored.

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 linkpath is denied, or one of the directories in the path prefix of linkpath did not allow search permission. (See also path_resolution(7).)

EDQUOT The user's quota of resources on the filesystem has been ex? hausted. The resources could be inodes or disk blocks, depend? ing on the filesystem implementation.

EEXIST linkpath already exists.

EFAULT target or linkpath points outside your accessible address space.

EIO An I/O error occurred.

ELOOP Too many symbolic links were encountered in resolving linkpath.

ENAMETOOLONG

target or linkpath was too long.

ENOENT A directory component in linkpath does not exist or is a dan?

gling symbolic link, or target or linkpath is an empty string.

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 linkpath is not, in fact, a

directory.

EPERM The filesystem containing linkpath does not support the creation

of symbolic links.

EROFS linkpath is on a read-only filesystem.

The following additional errors can occur for symlinkat():

EBADF newdirfd is not a valid file descriptor.

ENOENT linkpath is a relative pathname and newdirfd refers to a direc?

tory that has been deleted.

ENOTDIR

linkpath is relative and newdirfd is a file descriptor referring

to a file other than a directory.

VERSIONS

symlinkat() was added to Linux in kernel 2.6.16; library support was

added to glibc in version 2.4.

CONFORMING TO

symlink(): SVr4, 4.3BSD, POSIX.1-2001, POSIX.1-2008.

symlinkat(): POSIX.1-2008.

NOTES

No checking of target is done.

Deleting the name referred to by a symbolic link will actually delete

the file (unless it also has other hard links). If this behavior is not desired, use link(2).

Glibc notes

On older kernels where symlinkat() is unavailable, the glibc wrapper function falls back to the use of symlink(). When linkpath is a relative pathname, glibc constructs a pathname based on the symbolic link in /proc/self/fd that corresponds to the newdirfd argument.

SEE ALSO

ln(1), namei(1), lchown(2), link(2), lstat(2), open(2), readlink(2), rename(2), unlink(2), path_resolution(7), symlink(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/.