



Linux Ubuntu 26.04 Manual Pages on command 'symlink.2'

\$ man symlink.2

symlink(2) System Calls Manual symlink(2)

NAME

symlink, symlinkat - make a new name for a file

LIBRARY

Standard C library (libc, -lc)

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 <= 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 substi?

tuted 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 ignored when following the link (except when the `protected_symlinks` feature is enabled, as explained in `proc(5)`), 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 `symlink()`, 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.

See `openat(2)` for an explanation of the need for `symlinkat()`.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and `errno` is set to indicate the error.

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)`.)

EBADF (`symlinkat()`) `linkpath` is relative but `newdirfd` is neither `AT_FDCWD` nor a valid file descriptor.

EDQUOT The user's quota of resources on the filesystem has been exhausted. The resources could be inodes or disk blocks, depending 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 dangling symbolic link, or

target or linkpath is an empty string.

ENOENT (symlinkat()) linkpath is a relative pathname and newdirfd refers to a directory that has been deleted.

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.

ENOTDIR

(symlinkat()) linkpath is relative and newdirfd is a file descriptor referring to a file other than a directory.

EPERM The filesystem containing linkpath does not support the creation of symbolic links.

EROFS linkpath is on a read-only filesystem.

STANDARDS

POSIX.1-2024.

HISTORY

symlink()

SVr4, 4.3BSD, POSIX.1-2001.

symlinkat()

POSIX.1-2008. Linux 2.6.16, glibc 2.4.

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.

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).

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)

Linux man-pages 6.17

2026-02-08

symlink(2)