

symlink() — Create a Symbolic Link to a Pathname

Standards

Standards / Extensions	C or C++	Dependencies
POSIX.1a XPG4.2 Single UNIX Specification, Version 3	both	

Format

```
#define _POSIX1_SOURCE 2
#include <unistd.h>

int symlink(const char *pathname, const char *slink);
```

General Description

Creates the symbolic link named by *slink* with the file specified by *pathname*. File access checking is not performed on the file *pathname*, and the file need not exist. In addition, a symbolic link can cross file system boundaries.

A symbolic link pathname is resolved in this fashion:

- When a component of a pathname refers to a symbolic link rather than to a directory, the pathname contained in the symbolic link is resolved.
- If the pathname in the symbolic link begins with / (slash), the symbolic link pathname is resolved relative to the process root directory.

If the pathname in the symbolic link does not start with / (slash), the symbolic link pathname is resolved relative to the directory that contains the symbolic link.

- If the symbolic link is the last component of a pathname, it may or may not be resolved. Resolution depends on the function using the pathname. For example, `rename()` does not resolve a symbolic link when it appears as the final component of either the new or old pathname. However, `OPEN` does resolve a symbolic link when it appears as the last component.
- If the symbolic link is not the last component of the original pathname, remaining components of the original pathname are resolved relative to the symbolic link.
- When a / (slash) is the last component of a pathname and it is preceded by a symbolic link, the symbolic link is always resolved.

Because the mode of a symbolic link cannot be changed, its mode is ignored during the lookup process. Any files and directories to which a symbolic link refers are checked for access permission.

Returned Value

If successful, `symlink()` returns 0.

If unsuccessful, `symlink()` returns -1, does not affect any file it names, and sets `errno` to one of the following values:

Error Code

	Description
EACCES	A component of the <i>slink</i> path prefix denies search permission, or write permission is denied in the parent directory of the symbolic link to be created.
EEXIST	The file named by <i>slink</i> already exists.
EINVAL	This may be returned for either of these reasons:

- There is a NULL character in *pathname*.
- *slink* has a slash as its last component, which indicates that the preceding component will be a directory. A

symbolic link cannot be a directory.

EIO

Added for XPG4.2: An I/O error occurred while reading from the file system.

ELOOP

A loop exists in symbolic links. This error is issued if more than POSIX_SYMLINK symbolic links are encountered during resolution of the *slink* argument.

ENAMETOOLONG

pathname is longer than **PATH_MAX** characters, or some component of *pathname* is longer than **NAME_MAX** characters while **_POSIX_NO_TRUNC** is in effect. For symbolic links, the length of the pathname string substituted for a symbolic link exceeds **PATH_MAX**. The **PATH_MAX** and **NAME_MAX** values can be determined with `pathconf()`.

ENOENT

Added for XPG4.2: A component of *slink* does not name an existing file or *slink* is an empty string.

ENOSPC

The new symbolic link cannot be created because there is no space left on the file system that will contain the symbolic link.

ENOTDIR

A component of the path prefix of *slink* is not a directory.

EROFS

The file *slink* cannot reside on a read-only system.