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.3p' command

$ man symlink.3p

SYMLINK(3P) POSIX Programmer's Manual SYMLINK(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

symlink, symlinkat ? make a symbolic link

SYNOPSIS

```
#include <unistd.h>
int symlink(const char *path1, const char *path2);
#include <fcntl.h>
int symlinkat(const char *path1, int fd, const char *path2);
```

DESCRIPTION

The symlink() function shall create a symbolic link called path2 that contains the string pointed to by path1 (path2 is the name of the sym? bolic link created, path1 is the string contained in the symbolic link).

The string pointed to by path1 shall be treated only as a string and shall not be validated as a pathname.

If the symlink() function fails for any reason other than [EIO], any file named by path2 shall be unaffected.

If path2 names a symbolic link, symlink() shall fail and set errno to

[EEXIST].

The symbolic link's user ID shall be set to the process' effective user ID. The symbolic link's group ID shall be set to the group ID of the parent directory or to the effective group ID of the process. Implementations shall provide a way to initialize the symbolic link's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the symbolic link's group ID to the effective group ID of the calling process.

The values of the file mode bits for the created symbolic link are unspecified. All interfaces specified by POSIX.1?2008 shall behave as if the contents of symbolic links can always be read, except that the value of the file mode bits returned in the st_mode field of the stat structure is unspecified.

Upon successful completion, symlink() shall mark for update the last data access, last data modification, and last file status change timestamps of the symbolic link. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

The symlinkat() function shall be equivalent to the symlink() function except in the case where path2 specifies a relative path. In this case the symbolic link is created relative to the directory associated with the file descriptor fd instead of the current working directory. If the access mode of the open file description associated with the file descriptor is not O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the access mode is O_SEARCH, the function shall not perform the check.

If symlinkat() is passed the special value AT_FDCWD in the fd parameter, the current working directory shall be used and the behavior shall be identical to a call to symlink().

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set errno to indicate the error.

ERRORS

These functions shall fail if:

EACCES Write permission is denied in the directory where the symbolic link is being created, or search permission is denied for a com? ponent of the path prefix of path2.

EEXIST The path2 argument names an existing file.

EIO An I/O error occurs while reading from or writing to the file system.

ELOOP A loop exists in symbolic links encountered during resolution of the path2 argument.

ENAMETOOLONG

The length of a component of the pathname specified by the path2 argument is longer than {NAME_MAX} or the length of the path1 argument is longer than {SYMLINK_MAX}.

ENOENT A component of the path prefix of path2 does not name an exist? ing file or path2 is an empty string.

ENOENT or ENOTDIR

The path2 argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters. If path2 without the trailing <slash> characters would name an existing file, an [ENOENT] error shall not occur.

ENOSPC The directory in which the entry for the new symbolic link is being placed cannot be extended because no space is left on the file system containing the directory, or the new symbolic link cannot be created because no space is left on the file system which shall contain the link, or the file system is out of file-allocation resources.

ENOTDIR

A component of the path prefix of path2 names an existing file that is neither a directory nor a symbolic link to a directory.

EROFS The new symbolic link would reside on a read-only file system.

The symlinkat() function shall fail if:

EACCES The access mode of the open file description associated with fd

is not O_SEARCH and the permissions of the directory underlying fd do not permit directory searches.

EBADF The path2 argument does not specify an absolute path and the fd argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.

ENOTDIR

The path2 argument is not an absolute path and fd is a file de? scriptor associated with a non-directory file.

These functions may fail if:

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path2 argument.

ENAMETOOLONG

The length of the path2 argument exceeds {PATH_MAX} or pathname resolution of a symbolic link in the path2 argument produced an intermediate result with a length that exceeds {PATH_MAX}.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Like a hard link, a symbolic link allows a file to have multiple logi? cal names. The presence of a hard link guarantees the existence of a file, even after the original name has been removed. A symbolic link provides no such assurance; in fact, the file named by the path1 argu? ment need not exist when the link is created. A symbolic link can cross file system boundaries.

Normal permission checks are made on each component of the symbolic link pathname during its resolution.

RATIONALE

The purpose of the symlinkat() function is to create symbolic links in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to symlink(), resulting in unspecified behavior. By opening a file descriptor for the target directory and using the sym?

linkat() function it can be guaranteed that the created symbolic link is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

fdopendir(), fstatat(), lchown(), link(), open(), readlink(), rename(), unlink()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <unistd.h>

COPYRIGHT