NAME

rename – change the name of a file

SYNOPSIS

#include <stdio.h>

int rename(const char **old*, **const char** **new*);

MT-LEVEL

Async-Signal-Safe

DESCRIPTION

The function **rename( )** changes the name of a file. *old* points to the pathname of the file to be renamed. *new* points to the new pathname of the file.

If *old* and *new* both refer to the same existing file, the **rename( )** function returns successfully and performs no other action.

If *old* points to the pathname of a file that is not a directory, *new* must not point to the pathname of a directory. If the link named by *new* exists, it will be removed and *old* will be renamed to *new*. In this case, a link named *new* must remain visible to other processes throughout the renaming operation and will refer to either the file referred to by *new* or the file referred to as *old* before the operation began.

If *old* points to the pathname of a directory, *new* must not point to the pathname of a file that is not a directory. If the directory named by *new* exists, it will be removed and *old* will be renamed to *new*. In this case, a link named *new* will exist throughout the renaming operation and will refer to either the file referred to by *new* or the file referred to as *old* before the operation began. Thus, if *new* names an existing directory, it must be an empty directory.

The *new* pathname must not contain a path prefix that names *old*. Write access permission is required for both the directory containing *old* and the directory containing *new*. If *old* points to the pathname of a directory, write access permission is required for the directory named by *old*, and, if it exists, the directory named by *new*.

If the directory containing *old* has the sticky bit set, at least one of the following conditions listed below must be true:

- the user must own *old*
- the user must own the directory containing *old*
- *old* must be writable by the user
- the user must be a privileged user

If *new* exists, and the directory containing *new* is writable and has the sticky bit set, at least one of the following conditions must be true:

- the user must own *new*
- the user must own the directory containing *new*
- *new* must be writable by the user
- the user must be a privileged user

If the link named by *new* exists, the file's link count becomes zero when it is removed, and no process has the file open, then the space occupied by the file will be freed and the file will no longer be accessible. If one or more processes have the file open when the last link is removed, the link will be removed before **rename( )** returns, but the removal of the file contents will be postponed until all references to the file have been closed.

Upon successful completion, the **rename( )** function will mark for update the **st_ctime** and **st_mtime** fields of the parent directory of each file.

RETURN VALUES

Upon successful completion, the function **rename( )** returns a value of **0**; otherwise, it returns a value of −**1** and sets **errno** to indicate an error.

ERRORS

Under the following conditions, the function **rename( )** fails, and sets **errno** to:

EACCES A component of either path prefix denies search permission; one of the directories containing *old* and *new* denies write permissions; or write permission is denied by a directory pointed to by *old* or *new*.

EBUSY *new* is a directory and the mount point for a mounted file system.

EEXIST The link named by *new* is a directory containing entries other than '**.**' (the directory itself) and '**..**' (the parent directory).

EINVAL *new* directory pathname contains a path prefix that names the *old* directory.

EISDIR *new* points to a directory but *old* points to a file that is not a directory.

ELOOP Too many symbolic links were encountered in translating the pathname.

ENAMETOOLONG The length of *old* or *new* exceeds **{PATH_MAX},** or a pathname component is longer than **{NAME_MAX}** while **{_POSIX_NO_TRUNC}** is in effect.

EMLINK The file named by *old* is a directory, and the link count of the parent directory of *new* would exceed **{LINK_MAX}.**

ENOENT The link named by *old* does not exist, or either *old* or *new* points to an empty string.

ENOSPC The directory that would contain *new* cannot be extended.

ENOTDIR A component of either path prefix is not a directory, or *old* names a directory and *new* names a nondirectory file.

EROFS The requested operation requires writing in a directory on a read-only file system.

EXDEV The links named by *old* and *new* are on different file systems.

EIO An I/O error occurred while making or updating a directory entry.

SEE ALSO

chmod(2), **link**(2), **unlink**(2)

NOTES

The system can deadlock if there is a loop in the file system graph. Such a loop takes the form of an entry in directory *a*, say *a/name1*, being a hard link to directory *b*, and an entry in directory *b*, say *b/name2*, being a hard link to directory *a*. When such a loop exists and two separate processes attempt to rename *a/name1* to *b/name2* and rename *b/name2* to *a/name1*, respectively, the system may deadlock attempting to lock both directories for modification. Use symbolic links instead of hard links for directories.