rmdir() — Remove a Directory

Standards

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

Format

```
#define_POSIX_SOURCE
#include <unistd.h>

int rmdir(const char *pathname);
```

General Description

Removes a directory, *pathname*, provided that the directory is empty. *pathname* must not end in . (dot) or .. (dot-dot).

If *pathname* refers to a symbolic link, rmdir() does not affect any file or directory named by the contents of the symbolic link. rmdir() does not remove a directory that still contains files or subdirectories.

Special Behavior for XPG4.2

If *pathname* refers to a symbolic link, rmdir() fails and sets errno to ENOTDIR.

If no process currently has the directory open, rmdir() deletes the directory itself. The space occupied by the directory is freed for new use. If one or more processes have the directory open when it is removed, the directory itself is not removed until the last process closes the directory. New files cannot be created under a directory after the last link is removed, even if the directory is still open.

rmdir() removes the directory even if it is the working directory of a process.

If rmdir() is successful, the change and modification times for the parent directory are updated.

Returned Value

If successful, rmdir() returns 0.

If unsuccessful, rmdir() returns -1 and sets errno to one of the following values:

Error Code
: **Description**

EACCES
: The process did not have search permission for some component of *pathname*, or it did not have write permission for the directory containing the directory to be removed.

EBUSY
: *pathname* cannot be removed, because it is currently being used by the system or a process.

EINVAL
: The last component of *pathname* contains a . (dot) or a .. (dot-dot).

EIO
: **Added for XPG4.2:** A physical I/O error has occurred.

ELOOP
: A loop exists in symbolic links. More than POSIX_SYMLOOP (an integer defined in the limits.h header file) symbolic links are detected in the resolution of *pathname*.

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 using pathconf().

ENOENT

pathname does not exist, or it is an empty string.

ENOTDIR

Some component of the *pathname* prefix is not a directory.

ENOTEMPTY

The directory still contains files or subdirectories.

EPERM or EACCES

Added for XPG4.2: The S_ISVTX flag is set on the parent directory of the directory to be removed and the caller is not the owner of the directory to be removed, nor is the caller the owner of the parent directory, nor does the caller have the appropriate privileges.

EROFS

The directory to be removed is on a read-only file system.