NAME

unlink – remove directory entry

SYNOPSIS

#include <unistd.h>

int unlink(const char **path****);**

MT-LEVEL

Async-Signal-Safe

DESCRIPTION

unlink( ) removes the directory entry named by the path name pointed to by *path*. and decrements the link count of the file referenced by the directory entry. When all links to a file have been removed and no process has the file open, the space occupied by the file is freed and the file ceases to exist. If one or more processes have the file open when the last link is removed, space occupied by the file is not released until all references to the file have been closed. If *path* is a symbolic link, the symbolic link is removed. *path* should not name a directory unless the process has appropriate privileges. Applications should use **rmdir** to remove directories.

Upon successful completion **unlink( )** marks for update the **st_ctime** and **st_mtime** fields of the parent directory. Also, if the file's link count is not zero, the **st_ctime** field of the file is marked for update.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

The named file is unlinked unless one or more of the following are true:

EACCES	Search permission is denied for a component of the *path* prefix.
EACCES	Write permission is denied on the directory containing the link to be removed.
EACCES	The parent directory has the sticky bit set and the file is not writable by the user; the user does not own the parent directory and the user does not own the file.
EBUSY	The entry to be unlinked is the mount point for a mounted file system.
EFAULT	*path* points to an illegal address.
EINTR	A signal was caught during the **unlink( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {**PATH_MAX**}, or the length of a *path* component exceeds {**NAME_MAX**} while {**_POSIX_NO_TRUNC**} is in effect.
ENOENT	The named file does not exist or is a null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no

	longer active.
ENOTDIR	A component of the *path* prefix is not a directory.
EPERM	The named file is a directory and the effective user of the calling process is not super-user.
EROFS	The directory entry to be unlinked is part of a read-only file system.

SEE ALSO **rm**(1), **close**(2), **link**(2), **open**(2), **rmdir**(2)