NAME

umount, umount2 − unmount file system

SYNOPSIS

#include <sys/mount.h>

int umount(const char ****target*);**

int umount2(const char ****target*, int** *flags***);**

DESCRIPTION

umount() and **umount2**() remove the attachment of the (topmost) file system mounted on *target*.

Appropriate privilege (Linux: the **CAP_SYS_ADMIN** capability) is required to unmount file systems.

Linux 2.1.116 added the **umount2**() system call, which, like **umount**(), unmounts a target, but allows additional *flags* controlling the behavior of the operation:

MNT_FORCE (since Linux 2.1.116)
: Force unmount even if busy. This can cause data loss. (Only for NFS mounts.)

MNT_DETACH (since Linux 2.4.11)
: Perform a lazy unmount: make the mount point unavailable for new accesses, and actually perform the unmount when the mount point ceases to be busy.

MNT_EXPIRE (since Linux 2.6.8)
: Mark the mount point as expired. If a mount point is not currently in use, then an initial call to **umount2**() with this flag fails with the error **EAGAIN**, but marks the mount point as expired. The mount point remains expired as long as it isn't accessed by any process. A second **umount2**() call specifying **MNT_EXPIRE** unmounts an expired mount point. This flag cannot be specified with either **MNT_FORCE** or **MNT_DETACH**.

UMOUNT_NOFOLLOW (since Linux 2.6.34)
: Don't dereference *target* if it is a symbolic link. This flag allows security problems to be avoided in set-user-ID-*root* programs that allow unprivileged users to unmount file systems.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

The error values given below result from filesystem type independent errors. Each filesystem type may have its own special errors and its own special behavior. See the kernel source code for details.

EAGAIN
: A call to **umount2**() specifying **MNT_EXPIRE** successfully marked an unbusy file system as expired.

EBUSY
: *target* could not be unmounted because it is busy.

EFAULT
: *target* points outside the user address space.

EINVAL
: *target* is not a mount point. Or, **umount2**() was called with **MNT_EXPIRE** and either **MNT_DETACH** or **MNT_FORCE**.

ENAMETOOLONG
: A pathname was longer than **MAXPATHLEN**.

ENOENT
: A pathname was empty or had a nonexistent component.

ENOMEM
The kernel could not allocate a free page to copy filenames or data into.

EPERM
The caller does not have the required privileges.

CONFORMING TO

These functions are Linux-specific and should not be used in programs intended to be portable.

NOTES

The original **umount**() function was called as *umount(device)* and would return **ENOTBLK** when called with something other than a block device. In Linux 0.98p4 a call *umount(dir)* was added, in order to support anonymous devices. In Linux 2.3.99-pre7 the call *umount(device)* was removed, leaving only *umount(dir)* (since now devices can be mounted in more than one place, so specifying the device does not suffice).

SEE ALSO

mount(2), **path_resolution**(7), **mount**(8), **umount**(8)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.