



Full credit is given to the above companies including the OS that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'umount.2'

\$ man umount.2

UMOUNT(2) Linux Programmer's Manual UMOUNT(2)

NAME

umount, umount2 - unmount filesystem

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) filesystem mounted on target.

Appropriate privilege (Linux: the CAP_SYS_ADMIN capability) is required to unmount filesystems.

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)

Ask the filesystem to abort pending requests before attempting the unmount. This may allow the unmount to complete without

waiting for an inaccessible server, but could cause data loss.

If, after aborting requests, some processes still have active references to the filesystem, the unmount will still fail. As at Linux 4.12, MNT_FORCE is supported only on the following filesystems: 9p (since Linux 2.6.16), ceph (since Linux 2.6.34), cifs (since Linux 2.6.12), fuse (since Linux 2.6.16), lustre (since Linux 3.11), and NFS (since Linux 2.1.116).

MNT_DETACH (since Linux 2.4.11)

Perform a lazy unmount: make the mount point unavailable for new accesses, immediately disconnect the filesystem and all filesystems mounted below it from each other and from the mount table, 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 filesystems.

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 Linux kernel source code for details.

EAGAIN A call to `umount2()` specifying `MNT_EXPIRE` successfully marked an

unbusy filesystem 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.

EINVAL `umount2()` was called with `MNT_EXPIRE` and either `MNT_DETACH` or

`MNT_FORCE`.

EINVAL (since Linux 2.6.34)

`umount2()` was called with an invalid flag value in `flags`.

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.

VERSIONS

`MNT_DETACH` and `MNT_EXPIRE` are available in `glibc` since version 2.11.

CONFORMING TO

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

NOTES

`umount()` and shared mount points

Shared mount points cause any mount activity on a mount point, including `umount()` operations, to be forwarded to every shared mount point in the peer group and every slave mount of that peer group. This means that `umount()` of any peer in a set of shared mounts will cause all of its peers to be unmounted and all of their slaves to be unmounted as well.

This propagation of unmount activity can be particularly surprising on systems where every mount point is shared by default. On such systems, recursively bind mounting the root directory of the filesystem onto a subdirectory and then later unmounting that subdirectory with `MNT_DETACH` will cause every mount in the mount namespace to be lazily unmounted.

To ensure `umount()` does not propagate in this fashion, the mount point may be remounted using a `mount(2)` call with a `mount_flags` argument that includes both `MS_REC` and `MS_PRIVATE` prior to `umount()` being called.

Historical details

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)`, `mount_namespaces(7)`, `path_resolution(7)`, `mount(8)`, `umount(8)`

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.