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

Linux Ubuntu 26.04 Manual Pages on command 'statmount.2'

$ man statmount.2

statmount(2) System Calls Manual statmount(2)

NAME

statmount - get a mount status

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <linux/mount.h>  /* Definition of STATMOUNT_* constants */
#include <unistd.h>
int syscall(SYS_statmount, struct mnt_id_req *req,
            struct statmount *smbuf, size_t bufsize,
            unsigned long flags);
#include <linux/mount.h>
struct mnt_id_req {
  __u32  size;    /* sizeof(struct mnt_id_req) */
  __u64  mnt_id;  /* The mnt_id being queried */
  __u64  param;   /* An ORed combination of the STATMOUNT_ constants */
};
struct statmount {
  __u32  size;
  __u64  mask;
  __u32  sb_dev_major;
  __u32  sb_dev_minor;
  __u64  sb_magic;
```

```
    __u32  sb_flags;
    __u32  fs_type;
    __u64  mnt_id;
    __u64  mnt_parent_id;
    __u32  mnt_id_old;
    __u32  mnt_parent_id_old;
    __u64  mnt_attr;
    __u64  mnt_propagation;
    __u64  mnt_peer_group;
    __u64  mnt_master;
    __u64  propagate_from;
    __u32  mnt_root;
    __u32  mnt_point;
    char   str[];
};
```

Note: glibc provides no wrapper for statmount(), necessitating the use of syscall(2).

DESCRIPTION

To access a mount's status, the caller must have CAP_SYS_ADMIN in the user namespace.

This function returns information about a mount, storing it in the buffer pointed to by sm? buf. The returned buffer is a struct statmount which is of size bufsize with the fields filled in as described below. flags must be 0.

(Note that reserved space and padding is omitted.)

The mnt_id_req structure

req.size is used by the kernel to determine which struct mnt_id_req is being passed in; it should always be set to sizeof(struct mnt_id_req).

req.mnt_id can be obtained from either statx(2) using STATX_MNT_ID_UNIQUE or from list? mount(2) and is used as the identifier to query the status of the desired mount point.

req.param is used to tell the kernel which fields the caller is interested in. It is an ORed combination of the following constants

```
    STATMOUNT_SB_BASIC          /* Want/got sb_* */
    STATMOUNT_MNT_BASIC         /* Want/got mnt_* */
    STATMOUNT_PROPAGATE_FROM    /* Want/got propagate_from */
    STATMOUNT_MNT_ROOT          /* Want/got mnt_root  */
```

STATMOUNT_MNT_POINT /* Want/got mnt_point */

STATMOUNT_FS_TYPE /* Want/got fs_type */

In general, the kernel does not reject values in req.param other than the above. (For an exception, see EINVAL in errors.) Instead, it simply informs the caller which values are supported by this kernel and filesystem via the statmount.mask field. Therefore, do not simply set req.param to UINT_MAX (all bits set), as one or more bits may, in the future, be used to specify an extension to the buffer.

The returned information

The status information for the target mount is returned in the statmount structure pointed to by smbuf.

The fields in the statmount structure are:

smbuf.size

The size of the returned smbuf structure, including any of the strings fields that were filled.

smbuf.mask

The ORed combination of STATMOUNT_* flags indicating which fields were filled in and thus valid. The kernel may return fields that weren't requested, and may fail to re? turn fields that were requested, depending on what the backing file system and kernel supports. In either case, req.param will not be equal to mask.

smbuf.sb_dev_major

smbuf.sb_dev_minor

The device that is mounted at this mount point.

smbuf.sb_magic

The file system specific super block magic.

smbuf.sb_flags

The flags that are set on the super block, an ORed combination of SB_RDONLY, SB_SYN? CHRONOUS, SB_DIRSYNC, SB_LAZYTIME.

smbuf.fs_type

The offset to the location in the smbuf.str buffer that contains the string represen? tation of the mounted file system. It is a null-terminated string.

smbuf.mnt_id

The unique mount ID of the mount.

smbuf.mnt_parent_id

The unique mount ID of the parent mount point of this mount. If this is the root mount point then smbuf.mnt_id == smbuf.parent_mount_id.

smbuf.mnt_id_old

This corresponds to the mount ID that is exported by /proc/pid/mountinfo.

smbuf.mnt_parent_id_old

This corresponds to the parent mount ID that is exported by /proc/pid/mountinfo.

smbuf.mnt_attr

The MOUNT_ATTR_* flags set on this mount point.

smbuf.mnt_propagation

The mount propagation flags, which can be one of MS_SHARED, MS_SLAVE, MS_PRIVATE, MS_UNBINDABLE.

smbuf.mnt_peer_group

The ID of the shared peer group.

smbuf.mnt_master

The mount point receives its propagation from this mount ID.

smbuf.propagate_from

The ID from the namespace we propagated from.

smbuf.mnt_root

The offset to the location in the smbuf.str buffer that contains the string represen? tation of the mount relative to the root of the file system. It is a null-terminated string.

smbuf.mnt_point

The offset to the location in the smbuf.str buffer that contains the string represen? tation of the mount relative to the current root (ie if you are in a chroot). It is a null-terminated string.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set to indicate the error.

ERRORS

EPERM Permission is denied for accessing this mount.

EFAULT req or smbuf points to a location outside the process's accessible address space.

EINVAL Invalid flag specified in flags.

EINVAL req is of insufficient size to be utilized.

E2BIG req is too large.

EOVERFLOW

The size of smbuf is too small to contain either the smbuf.fs_type, smbuf.mnt_root, or smbuf.mnt_point. Allocate a larger buffer and retry the call.

ENOENT The specified req.mnt_id doesn't exist.

ENOMEM Out of memory (i.e., kernel memory).

STANDARDS

Linux.

SEE ALSO

listmount(2), statx(2)