mount(8) — Linux manual page

NAME | SYNOPSIS | DESCRIPTION | COMMAND-LINE OPTIONS | FILESYSTEM-INDEPENDENT MOUNT OPTIONS | FILESYSTEM-SPECIFIC MOUNT OPTIONS | DM-VERITY SUPPORT | LOOP-DEVICE SUPPORT | EXIT STATUS | EXTERNAL HELPERS | ENVIRONMENT | FILES | HISTORY | BUGS | AUTHORS | SEE ALSO | REPORTING BUGS | AVAILABILITY

NAME top

mount - mount a filesystem

SYNOPSIS top

mount [**-h**|**-V**]

mount [**-l**] [**-t** *fstype*]

mount -a [**-fFnrsvw**] [**-t** *fstype*] [**-O** *optlist*]

mount [**-fnrsvw**] [**-o** *options*] *device*|*mountpoint*

mount [**-fnrsvw**] [**-t** *fstype*] [**-o** *options*] *device mountpoint*

mount --bind|**--rbind**|**--move** *olddir newdir*

mount
--make-[**shared**|**slave**|**private**|**unbindable**|**rshared**|**rslave**|**rprivate**|**runbindable**]
mountpoint

DESCRIPTION top

All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The **mount** command serves to attach the filesystem found on some device to the big file tree. Conversely, the umount(8) command will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by network or other services.

The standard form of the **mount** command is:

mount -t *type device dir*

This tells the kernel to attach the filesystem found on *device* (which is of type *type*) at the directory *dir*. The option **-t** *type* is optional. The **mount** command is usually able to detect a filesystem. The root permissions are necessary to mount a filesystem by default. See section "Non-superuser mounts" below for more details. The previous contents (if any) and owner and mode of *dir* become invisible, and as long as this filesystem remains mounted, the pathname *dir* refers to the root of the filesystem on *device*.

If only the directory or the device is given, for example:

```
mount /dir
```

then **mount** looks for a mountpoint (and if not found then for a device) in the */etc/fstab* file. It’s possible to use the **--target** or **--source** options to avoid ambiguous interpretation of the given argument. For example:

```
mount --target /mountpoint
```

The same filesystem may be mounted more than once, and in some cases (e.g., network filesystems) the same filesystem may be mounted on the same mountpoint multiple times. The **mount** command does not implement any policy to control this behavior. All behavior is controlled by the kernel and it is usually specific to the filesystem driver. The exception is **--all**, in this case already mounted filesystems are ignored (see **--all** below for more details).

Shared subtree operations

Since Linux 2.6.15 it is possible to mark a mount and its submounts as shared, private, slave or unbindable. A shared mount provides the ability to create mirrors of that mount such that mounts and unmounts within any of the mirrors propagate to the other mirror. A slave mount receives propagation from its master, but not vice versa. A private mount carries no propagation abilities. An unbindable mount is a private mount which cannot be cloned through a bind operation. The detailed semantics are documented in *Documentation/filesystems/sharedsubtree.txt* file in the kernel source tree; see also mount_namespaces(7).

Supported operations are:

```
mount --make-shared mountpoint
mount --make-slave mountpoint
mount --make-private mountpoint
mount --make-unbindable mountpoint
```

The following commands allow one to recursively change the type of all the mounts under a given mountpoint.

```
mount --make-rshared mountpoint
mount --make-rslave mountpoint
mount --make-rprivate mountpoint
mount --make-runbindable mountpoint
```

mount(8) **does not read fstab**(5) when a **--make-*** operation is requested. All necessary information has to be specified on the command line.

Note that the Linux kernel does not allow changing multiple propagation flags with a single mount(2) system call, and the flags cannot be mixed with other mount options and operations.

Since util-linux 2.23 the **mount** command can be used to do more propagation (topology) changes by one mount(8) call and do it also together with other mount operations. The propagation flags are applied by additional mount(2) system calls when the preceding mount operations were successful. Note that this use case is not atomic. It is possible to specify the propagation flags in fstab(5) as mount options (**private**, **slave**, **shared**, **unbindable**, **rprivate**, **rslave**, **rshared**, **runbindable**).

For example:

```
    mount --make-private --make-unbindable /dev/sda1 /foo
```

is the same as:

```
    mount /dev/sda1 /foo
    mount --make-private /foo
    mount --make-unbindable /foo
```

FILESYSTEM-INDEPENDENT MOUNT OPTIONS top

Some of these options are only useful when they appear in the */etc/fstab* file.

Some of these options could be enabled or disabled by default in the system kernel. To check the current setting see the options in */proc/mounts*. Note that filesystems also have per-filesystem specific default mount options (see for example **tune2fs -l** output for ext*N* filesystems).

The following options apply to any filesystem that is being mounted (but not every filesystem actually honors them - e.g., the **sync** option today has an effect only for ext2, ext3, ext4, fat, vfat, ufs and xfs):

async
All I/O to the filesystem should be done asynchronously. (See also the **sync** option.)

atime
Do not use the **noatime** feature, so the inode access time is controlled by kernel defaults. See also the descriptions of the **relatime** and **strictatime** mount options.

noatime
Do not update inode access times on this filesystem (e.g. for faster access on the news spool to speed up news servers). This works for all inode types (directories too), so it implies **nodiratime**.

auto
Can be mounted with the **-a** option.

noauto
Can only be mounted explicitly (i.e., the **-a** option will not cause the filesystem to be mounted).

context=*context*, **fscontext=***context*, **defcontext=***context*, and **rootcontext=***context*
The **context=** option is useful when mounting filesystems that do not support extended attributes, such as a floppy or hard disk formatted with VFAT, or systems that are not normally running under SELinux, such as an ext3 or ext4 formatted disk

from a non-SELinux workstation. You can also use **context=** on filesystems you do not trust, such as a floppy. It also helps in compatibility with xattr-supporting filesystems on earlier 2.4.<x> kernel versions. Even where xattrs are supported, you can save time not having to label every file by assigning the entire disk one security context.

A commonly used option for removable media is **context="system_u:object_r:removable_t**.

The **fscontext=** option works for all filesystems, regardless of their xattr support. The fscontext option sets the overarching filesystem label to a specific security context. This filesystem label is separate from the individual labels on the files. It represents the entire filesystem for certain kinds of permission checks, such as during mount or file creation. Individual file labels are still obtained from the xattrs on the files themselves. The context option actually sets the aggregate context that fscontext provides, in addition to supplying the same label for individual files.

You can set the default security context for unlabeled files using **defcontext=** option. This overrides the value set for unlabeled files in the policy and requires a filesystem that supports xattr labeling.

The **rootcontext=** option allows you to explicitly label the root inode of a FS being mounted before that FS or inode becomes visible to userspace. This was found to be useful for things like stateless Linux.

Note that the kernel rejects any remount request that includes the context option, **even** when unchanged from the current context.

Warning: the *context* value might contain commas, in which case the value has to be properly quoted, otherwise **mount** will interpret the comma as a separator between mount options. Don't forget that the shell strips off quotes and thus **double quoting is required**. For example:

```
mount -t tmpfs none /mnt -o \
'context="system_u:object_r:tmp_t:s0:c127,c456",noexec'
```

For more details, see selinux(8).

defaults

Use the default options: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, and **async**.

Note that the real set of all default mount options depends on the kernel and filesystem type. See the beginning of this section for more details.

dev

Interpret character or block special devices on the filesystem.

nodev

Do not interpret character or block special devices on the filesystem.

diratime

Update directory inode access times on this filesystem. This is the default. (This option is ignored when **noatime** is set.)

nodiratime
Do not update directory inode access times on this filesystem. (This option is implied when **noatime** is set.)

dirsync
All directory updates within the filesystem should be done synchronously. This affects the following system calls: creat(2), link(2), unlink(2), symlink(2), mkdir(2), rmdir(2), mknod(2) and rename(2).

exec
Permit execution of binaries.

noexec
Do not permit direct execution of any binaries on the mounted filesystem.

group
Allow an ordinary user to mount the filesystem if one of that user's groups matches the group of the device. This option implies the options **nosuid** and **nodev** (unless overridden by subsequent options, as in the option line **group**,**dev**,**suid**).

iversion
Every time the inode is modified, the i_version field will be incremented.

noiversion
Do not increment the i_version inode field.

mand
Allow mandatory locks on this filesystem. See fcntl(2).

nomand
Do not allow mandatory locks on this filesystem.

_netdev
The filesystem resides on a device that requires network access (used to prevent the system from attempting to mount these filesystems until the network has been enabled on the system).

nofail
Do not report errors for this device if it does not exist.

relatime
Update inode access times relative to modify or change time. Access time is only updated if the previous access time was earlier than the current modify or change time. (Similar to **noatime**, but it doesn't break **mutt**(1) or other applications that need to know if a file has been read since the last time it was modified.)

Since Linux 2.6.30, the kernel defaults to the behavior provided by this option (unless **noatime** was specified), and the **strictatime** option is required to obtain traditional semantics. In addition, since Linux 2.6.30, the file's last access time is always updated if it is more than 1 day old.

norelatime

Do not use the **relatime** feature. See also the **strictatime** mount option.

strictatime
Allows to explicitly request full atime updates. This makes it possible for the kernel to default to **relatime** or **noatime** but still allow userspace to override it. For more details about the default system mount options see */proc/mounts*.

nostrictatime
Use the kernel's default behavior for inode access time updates.

lazytime
Only update times (atime, mtime, ctime) on the in-memory version of the file inode.

This mount option significantly reduces writes to the inode table for workloads that perform frequent random writes to preallocated files.

The on-disk timestamps are updated only when:

- the inode needs to be updated for some change unrelated to file timestamps
- the application employs fsync(2), syncfs(2), or sync(2)
- an undeleted inode is evicted from memory
- more than 24 hours have passed since the inode was written to disk.

nolazytime
Do not use the lazytime feature.

suid
Honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem.

nosuid
Do not honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem. In addition, SELinux domain transitions require permission nosuid_transition, which in turn needs also policy capability nnp_nosuid_transition.

silent
Turn on the silent flag.

loud
Turn off the silent flag.

owner
Allow an ordinary user to mount the filesystem if that user is the owner of the device. This option implies the options **nosuid** and **nodev** (unless overridden by subsequent options, as in the option line **owner,dev,suid**).

remount
Attempt to remount an already-mounted filesystem. This is commonly used to change the mount flags for a filesystem, especially to make a readonly filesystem writable. It does

not change device or mount point.

The remount operation together with the **bind** flag has special semantics. See above, the subsection **Bind mounts**.

The remount functionality follows the standard way the **mount** command works with options from *fstab*. This means that **mount** does not read *fstab* (or *mtab*) only when both *device* and *dir* are specified.

mount -o remount,rw /dev/foo /dir

After this call all old mount options are replaced and arbitrary stuff from *fstab* (or *mtab*) is ignored, except the loop= option which is internally generated and maintained by the mount command.

mount -o remount,rw /dir

After this call, mount reads *fstab* and merges these options with the options from the command line (**-o**). If no mountpoint is found in *fstab*, then a remount with unspecified source is allowed.

mount allows the use of **--all** to remount all already mounted filesystems which match a specified filter (**-O** and **-t**). For example:

mount --all -o remount,ro -t vfat

remounts all already mounted vfat filesystems in read-only mode. Each of the filesystems is remounted by **mount -o remount,ro /dir** semantic. This means the **mount** command reads *fstab* or *mtab* and merges these options with the options from the command line.

ro
Mount the filesystem read-only.

rw
Mount the filesystem read-write.

sync
All I/O to the filesystem should be done synchronously. In the case of media with a limited number of write cycles (e.g. some flash drives), **sync** may cause life-cycle shortening.

user
Allow an ordinary user to mount the filesystem. The name of the mounting user is written to the *mtab* file (or to the private libmount file in */run/mount* on systems without a regular *mtab*) so that this same user can unmount the filesystem again. This option implies the options **noexec**, **nosuid**, and **nodev** (unless overridden by subsequent options, as in the option line **user,exec,dev,suid**).

nouser
Forbid an ordinary user to mount the filesystem. This is the default; it does not imply any other options.

users
Allow any user to mount and to unmount the filesystem, even when some other ordinary user mounted it. This option implies

the options **noexec**, **nosuid**, and **nodev** (unless overridden by subsequent options, as in the option line **users**,**exec**,**dev**,**suid**).

X-*
All options prefixed with "X-" are interpreted as comments or as userspace application-specific options. These options are not stored in user space (e.g., *mtab* file), nor sent to the mount.*type* helpers nor to the mount(2) system call. The suggested format is **X-***appname*.*option*.

x-*
The same as **X-*** options, but stored permanently in user space. This means the options are also available for umount(8) or other operations. Note that maintaining mount options in user space is tricky, because it's necessary use libmount-based tools and there is no guarantee that the options will be always available (for example after a move mount operation or in unshared namespace).

Note that before util-linux v2.30 the x-* options have not been maintained by libmount and stored in user space (functionality was the same as for X-* now), but due to the growing number of use-cases (in initrd, systemd etc.) the functionality has been extended to keep existing *fstab* configurations usable without a change.

X-mount.mkdir[=*mode*]
Allow to make a target directory (mountpoint) if it does not exist yet. The optional argument *mode* specifies the filesystem access mode used for mkdir(2) in octal notation. The default mode is 0755. This functionality is supported only for root users or when mount executed without suid permissions. The option is also supported as x-mount.mkdir, this notation is deprecated since v2.30. See also **--mkdir** command line option.

X-mount.subdir=*directory*
Allow mounting sub-directory from a filesystem instead of the root directory. For now, this feature is implemented by temporary filesystem root directory mount in unshared namespace and then bind the sub-directory to the final mount point and umount the root of the filesystem. The sub-directory mount shows up atomically for the rest of the system although it is implemented by multiple mount(2) syscalls. This feature is EXPERIMENTAL.

nosymfollow
Do not follow symlinks when resolving paths. Symlinks can still be created, and readlink(1), readlink(2), realpath(1), and realpath(3) all still work properly.

AUTHORS top

Karel Zak <kzak@redhat.com>

SEE ALSO top

mount(2), umount(2), filesystems(5), fstab(5), nfs(5), xfs(5), mount_namespaces(7), xattr(7), e2label(8), findmnt(8), losetup(8), lsblk(8), mke2fs(8), mountd(8), nfsd(8), swapon(8), tune2fs(8), umount(8), xfs_admin(8)

REPORTING BUGS top

For bug reports, use the issue tracker at https://github.com/karelzak/util-linux/issues.

AVAILABILITY top

The **mount** command is part of the util-linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util-linux/>. This page is part of the *util-linux* (a random collection of Linux utilities) project. Information about the project can be found at ⟨https://www.kernel.org/pub/linux/utils/util-linux/⟩. If you have a bug report for this manual page, send it to util-linux@vger.kernel.org. This page was obtained from the project's upstream Git repository ⟨git://git.kernel.org/pub/scm/utils/util-linux/util-linux.git⟩ on 2021-08-27. (At that time, the date of the most recent commit that was found in the repository was 2021-08-24.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is *not* part of the original manual page), send a mail to man-pages@man7.org