mount(8) — Linux manual page

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).

Listing the mounts

The listing mode is maintained for backward compatibility only.

For more robust and customizable output use findmnt(8), **especially in your scripts**. Note that control characters in the mountpoint name are replaced with '?'.

The following command lists all mounted filesystems (of type *type*):

mount [**-l**] [**-t** *type*]

The option **-l** adds labels to this listing. See below.

Indicating the device and filesystem

Most devices are indicated by a filename (of a block special device), like */dev/sda1*, but there are other possibilities. For example, in the case of an NFS mount, *device* may look like *knuth.cwi.nl:/dir*.

The device names of disk partitions are unstable; hardware reconfiguration, and adding or removing a device can cause changes in names. This is the reason why it's strongly recommended to use filesystem or partition identifiers like UUID or LABEL. Currently supported identifiers (tags):

LABEL=*label*
: Human readable filesystem identifier. See also **-L**.

UUID=*uuid*
: Filesystem universally unique identifier. The format of the UUID is usually a series of hex digits separated by hyphens. See also **-U**.

  Note that **mount** uses UUIDs as strings. The UUIDs from the command line or from fstab(5) are not converted to internal binary representation. The string representation of the UUID

should be based on lower case characters.

PARTLABEL=*label*
Human readable partition identifier. This identifier is independent on filesystem and does not change by mkfs or mkswap operations. It's supported for example for GUID Partition Tables (GPT).

PARTUUID=*uuid*
Partition universally unique identifier. This identifier is independent on filesystem and does not change by mkfs or mkswap operations. It's supported for example for GUID Partition Tables (GPT).

ID=*id*
Hardware block device ID as generated by udevd. This identifier is usually based on WWN (unique storage identifier) and assigned by the hardware manufacturer. See **ls /dev/disk/by-id** for more details, this directory and running udevd is required. This identifier is not recommended for generic use as the identifier is not strictly defined and it depends on udev, udev rules and hardware.

The command **lsblk --fs** provides an overview of filesystems, LABELs and UUIDs on available block devices. The command **blkid -p <device>** provides details about a filesystem on the specified device.

Don't forget that there is no guarantee that UUIDs and labels are really unique, especially if you move, share or copy the device. Use **lsblk -o +UUID,PARTUUID** to verify that the UUIDs are really unique in your system.

The recommended setup is to use tags (e.g. **UUID**=*uuid*) rather than */dev/disk/by-{label,uuid,id,partuuid,partlabel}* udev symlinks in the */etc/fstab* file. Tags are more readable, robust and portable. The mount(8) command internally uses udev symlinks, so the use of symlinks in */etc/fstab* has no advantage over tags. For more details see libblkid(3).

The *proc* filesystem is not associated with a special device, and when mounting it, an arbitrary keyword - for example, *proc* - can be used instead of a device specification. (The customary choice *none* is less fortunate: the error message 'none already mounted' from **mount** can be confusing.)

The files /etc/fstab, /etc/mtab and /proc/mounts

The file */etc/fstab* (see fstab(5)), may contain lines describing what devices are usually mounted where, using which options. The default location of the fstab(5) file can be overridden with the **--fstab** *path* command-line option (see below for more details).

The command

mount -a [**-t** *type*] [**-O** *optlist*]

(usually given in a bootscript) causes all filesystems mentioned in *fstab* (of the proper type and/or having or not having the proper options) to be mounted as indicated, except for those whose line contains the **noauto** keyword. Adding the **-F** option will make **mount** fork, so that the filesystems are mounted in parallel.

When mounting a filesystem mentioned in *fstab* or *mtab*, it

suffices to specify on the command line only the device, or only the mount point.

The programs **mount** and umount(8) traditionally maintained a list of currently mounted filesystems in the file */etc/mtab*. The support for regular classic */etc/mtab* is completely disabled at compile time by default, because on current Linux systems it is better to make */etc/mtab* a symlink to */proc/mounts* instead. The regular *mtab* file maintained in userspace cannot reliably work with namespaces, containers and other advanced Linux features. If the regular *mtab* support is enabled, then it's possible to use the file as well as the symlink.

If no arguments are given to **mount**, the list of mounted filesystems is printed.

If you want to override mount options from */etc/fstab*, you have to use the **-o** option:

mount *device*|*dir* **-o** *options*

and then the mount options from the command line will be appended to the list of options from */etc/fstab*. This default behaviour can be changed using the **--options-mode** command-line option. The usual behavior is that the last option wins if there are conflicting ones.

The **mount** program does not read the */etc/fstab* file if both *device* (or LABEL, UUID, ID, PARTUUID or PARTLABEL) and *dir* are specified. For example, to mount device **foo** at **/dir**:

mount /dev/foo /dir

This default behaviour can be changed by using the **--options-source-force** command-line option to always read configuration from *fstab*. For non-root users **mount** always reads the *fstab* configuration.

Non-superuser mounts

Normally, only the superuser can mount filesystems. However, when *fstab* contains the **user** option on a line, anybody can mount the corresponding filesystem.

Thus, given a line

/dev/cdrom /cd iso9660 ro,user,noauto,unhide

any user can mount the iso9660 filesystem found on an inserted CDROM using the command:

mount /cd

Note that **mount** is very strict about non-root users and all paths specified on command line are verified before *fstab* is parsed or a helper program is executed. It's strongly recommended to use a valid mountpoint to specify filesystem, otherwise **mount** may fail. For example it's a bad idea to use NFS or CIFS source on command line.

Since util-linux 2.35, **mount** does not exit when user permissions are inadequate according to libmount's internal security rules. Instead, it drops suid permissions and continues as regular non-root user. This behavior supports use-cases where root

permissions are not necessary (e.g., fuse filesystems, user namespaces, etc).

For more details, see fstab(5). Only the user that mounted a filesystem can unmount it again. If any user should be able to unmount it, then use **users** instead of **user** in the *fstab* line. The **owner** option is similar to the **user** option, with the restriction that the user must be the owner of the special file. This may be useful e.g. for */dev/fd* if a login script makes the console user owner of this device. The **group** option is similar, with the restriction that the user must be a member of the group of the special file.

Bind mount operation

Remount part of the file hierarchy somewhere else. The call is:

mount --bind *olddir newdir*

or by using this *fstab* entry:

/*olddir* **/***newdir* **none bind**

After this call the same contents are accessible in two places.

It is important to understand that "bind" does not create any second-class or special node in the kernel VFS. The "bind" is just another operation to attach a filesystem. There is nowhere stored information that the filesystem has been attached by a "bind" operation. The *olddir* and *newdir* are independent and the *olddir* may be unmounted.

One can also remount a single file (on a single file). It's also possible to use a bind mount to create a mountpoint from a regular directory, for example:

mount --bind foo foo

The bind mount call attaches only (part of) a single filesystem, not possible submounts. The entire file hierarchy including submounts can be attached a second place by using:

mount --rbind *olddir newdir*

Note that the filesystem mount options maintained by the kernel will remain the same as those on the original mount point. The userspace mount options (e.g., _netdev) will not be copied by **mount** and it's necessary to explicitly specify the options on the **mount** command line.

Since util-linux 2.27 **mount** permits changing the mount options by passing the relevant options along with **--bind**. For example:

mount -o bind,ro foo foo

This feature is not supported by the Linux kernel; it is implemented in userspace by an additional mount(2) remounting system call. This solution is not atomic.

The alternative (classic) way to create a read-only bind mount is to use the remount operation, for example:

mount --bind *olddir newdir* **mount -o remount,bind,ro** *olddir newdir*

Note that a read-only bind will create a read-only mountpoint (VFS entry), but the original filesystem superblock will still be writable, meaning that the *olddir* will be writable, but the *newdir* will be read-only.

It's also possible to change nosuid, nodev, noexec, noatime, nodiratime and relatime VFS entry flags via a "remount,bind" operation. The other flags (for example filesystem-specific flags) are silently ignored. It's impossible to change mount options recursively (for example with **-o rbind,ro**).

Since util-linux 2.31, **mount** ignores the **bind** flag from */etc/fstab* on a **remount** operation (if "-o remount" is specified on command line). This is necessary to fully control mount options on remount by command line. In previous versions the bind flag has been always applied and it was impossible to re-define mount options without interaction with the bind semantic. This **mount** behavior does not affect situations when "remount,bind" is specified in the */etc/fstab* file.

The move operation

Move a **mounted tree** to another place (atomically). The call is:

mount --move *olddir newdir*

This will cause the contents which previously appeared under *olddir* to now be accessible under *newdir*. The physical location of the files is not changed. Note that *olddir* has to be a mountpoint.

Note also that moving a mount residing under a shared mount is invalid and unsupported. Use **findmnt -o TARGET,PROPAGATION** to see the current propagation flags.

COMMAND-LINE OPTIONS [top]

The full set of mount options used by an invocation of **mount** is determined by first extracting the mount options for the filesystem from the *fstab* table, then applying any options specified by the **-o** argument, and finally applying a **-r** or **-w** option, when present.

The **mount** command does not pass all command-line options to the **/sbin/mount.***suffix* mount helpers. The interface between **mount** and the mount helpers is described below in the section **EXTERNAL HELPERS**.

Command-line options available for the **mount** command are:

-a, --all

Mount all filesystems (of the given types) mentioned in *fstab* (except for those whose line contains the **noauto** keyword). The filesystems are mounted following their order in *fstab*. The **mount** command compares filesystem source, target (and fs root for bind mount or btrfs) to detect already mounted filesystems. The kernel table with already mounted filesystems is cached during **mount --all**. This means that all duplicated *fstab* entries will be mounted.

The option **--all** is possible to use for remount operation too. In this case all filters (**-t** and **-O**) are applied to the

table of already mounted filesystems.

Since version 2.35 is possible to use the command line option **-o** to alter mount options from *fstab* (see also **--options-mode**).

Note that it is a bad practice to use **mount -a** for *fstab* checking. The recommended solution is **findmnt --verify**.

-B, --bind
Remount a subtree somewhere else (so that its contents are available in both places). See above, under **Bind mounts**.

-c, --no-canonicalize
Don't canonicalize paths. The **mount** command canonicalizes all paths (from the command line or *fstab*) by default. This option can be used together with the **-f** flag for already canonicalized absolute paths. The option is designed for mount helpers which call **mount -i**. It is strongly recommended to not use this command-line option for normal mount operations.

Note that **mount** does not pass this option to the **/sbin/mount.***type* helpers.

-F, --fork
(Used in conjunction with **-a**.) Fork off a new incarnation of **mount** for each device. This will do the mounts on different devices or different NFS servers in parallel. This has the advantage that it is faster; also NFS timeouts proceed in parallel. A disadvantage is that the order of the mount operations is undefined. Thus, you cannot use this option if you want to mount both */usr* and */usr/spool*.

-f, --fake
Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the filesystem. This option is useful in conjunction with the **-v** flag to determine what the **mount** command is trying to do. It can also be used to add entries for devices that were mounted earlier with the **-n** option. The **-f** option checks for an existing record in */etc/mtab* and fails when the record already exists (with a regular non-fake mount, this check is done by the kernel).

-i, --internal-only
Don't call the **/sbin/mount.***filesystem* helper even if it exists.

-L, --label *label*
Mount the partition that has the specified *label*.

-l, --show-labels
Add the labels in the mount output. **mount** must have permission to read the disk device (e.g. be set-user-ID root) for this to work. One can set such a label for ext2, ext3 or ext4 using the e2label(8) utility, or for XFS using xfs_admin(8), or for reiserfs using **reiserfstune**(8).

-M, --move
Move a subtree to some other place. See above, the subsection **The move operation**.

-m, --mkdir[*=mode*]
Allow to make a target directory (mountpoint) if it does not exist yet. Alias to "-o X-mount.mkdir[=mode]", the default mode is 0755. For more details see **X-mount.mkdir** below.

-n, --no-mtab
Mount without writing in */etc/mtab*. This is necessary for example when */etc* is on a read-only filesystem.

-N, --namespace *ns*
Perform the mount operation in the mount namespace specified by *ns*. *ns* is either PID of process running in that namespace or special file representing that namespace.

mount switches to the mount namespace when it reads */etc/fstab*, writes */etc/mtab: (or writes to _/run/mount*) and calls the mount(2) system call, otherwise it runs in the original mount namespace. This means that the target namespace does not have to contain any libraries or other requirements necessary to execute the mount(2) call.

See mount_namespaces(7) for more information.

-O, --test-opts *opts*
Limit the set of filesystems to which the **-a** option applies. In this regard it is like the **-t** option except that **-O** is useless without **-a**. For example, the command

mount -a -O no_netdev

mounts all filesystems except those which have the option *netdev* specified in the options field in the */etc/fstab* file.

It is different from **-t** in that each option is matched exactly; a leading **no** at the beginning of one option does not negate the rest.

The **-t** and **-O** options are cumulative in effect; that is, the command

mount -a -t ext2 -O _netdev

mounts all ext2 filesystems with the _netdev option, not all filesystems that are either ext2 or have the _netdev option specified.

-o, --options *opts*
Use the specified mount options. The *opts* argument is a comma-separated list. For example:

mount LABEL=mydisk -o noatime,nodev,nosuid

For more details, see the **FILESYSTEM-INDEPENDENT MOUNT OPTIONS** and **FILESYSTEM-SPECIFIC MOUNT OPTIONS** sections.

--options-mode *mode*
Controls how to combine options from *fstab*/*mtab* with options from the command line. *mode* can be one of **ignore**, **append**, **prepend** or **replace**. For example, **append** means that options from *fstab* are appended to options from the command line. The default value is **prepend** — it means command line options are evaluated after *fstab* options. Note that the last option wins if there are conflicting ones.

--options-source *source*
Source of default options. *source* is a comma-separated list of **fstab**, **mtab** and **disable**. **disable** disables **fstab** and **mtab** and disables **--options-source-force**. The default value is **fstab,mtab**.

--options-source-force
Use options from *fstab*/*mtab* even if both *device* and *dir* are specified.

-R, **--rbind**
Remount a subtree and all possible submounts somewhere else (so that its contents are available in both places). See above, the subsection **Bind mounts**.

-r, **--read-only**
Mount the filesystem read-only. A synonym is **-o ro**.

Note that, depending on the filesystem type, state and kernel behavior, the system may still write to the device. For example, ext3 and ext4 will replay the journal if the filesystem is dirty. To prevent this kind of write access, you may want to mount an ext3 or ext4 filesystem with the **ro,noload** mount options or set the block device itself to read-only mode, see the blockdev(8) command.

-s
Tolerate sloppy mount options rather than failing. This will ignore mount options not supported by a filesystem type. Not all filesystems support this option. Currently it's supported by the **mount.nfs** mount helper only.

--source *device*
If only one argument for the **mount** command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is the mount source.

--target *directory*
If only one argument for the mount command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is the mount target.

--target-prefix *directory*
Prepend the specified directory to all mount targets. This option can be used to follow *fstab*, but mount operations are done in another place, for example:

mount --all --target-prefix /chroot -o X-mount.mkdir

mounts all from system *fstab* to */chroot*, all missing mountpoint are created (due to X-mount.mkdir). See also **--fstab** to use an alternative *fstab*.

-T, **--fstab** *path*
Specifies an alternative *fstab* file. If *path* is a directory, then the files in the directory are sorted by strverscmp(3); files that start with "." or without an *.fstab* extension are ignored. The option can be specified more than once. This option is mostly designed for initramfs or chroot scripts where additional configuration is specified beyond standard

system configuration.

Note that **mount** does not pass the option **--fstab** to the **/sbin/mount.***type* helpers, meaning that the alternative *fstab* files will be invisible for the helpers. This is no problem for normal mounts, but user (non-root) mounts always require *fstab* to verify the user's rights.

-t, --types *fstype*
The argument following the **-t** is used to indicate the filesystem type. The filesystem types which are currently supported depend on the running kernel. See */proc/filesystems* and */lib/modules/$(uname -r)/kernel/fs* for a complete list of the filesystems. The most common are ext2, ext3, ext4, xfs, btrfs, vfat, sysfs, proc, nfs and cifs.

The programs **mount** and umount(8) support filesystem subtypes. The subtype is defined by a '.subtype' suffix. For example 'fuse.sshfs'. It's recommended to use subtype notation rather than add any prefix to the mount source (for example 'sshfs#example.com' is deprecated).

If no **-t** option is given, or if the **auto** type is specified, **mount** will try to guess the desired type. **mount** uses the libblkid(3) library for guessing the filesystem type; if that does not turn up anything that looks familiar, **mount** will try to read the file */etc/filesystems*, or, if that does not exist, */proc/filesystems*. All of the filesystem types listed there will be tried, except for those that are labeled "nodev" (e.g. *devpts*, *proc* and *nfs*). If */etc/filesystems* ends in a line with a single *, mount will read */proc/filesystems* afterwards. While trying, all filesystem types will be mounted with the mount option **silent**.

The **auto** type may be useful for user-mounted floppies. Creating a file */etc/filesystems* can be useful to change the probe order (e.g., to try vfat before msdos or ext3 before ext2) or if you use a kernel module autoloader.

More than one type may be specified in a comma-separated list, for the **-t** option as well as in an */etc/fstab* entry. The list of filesystem types for the **-t** option can be prefixed with **no** to specify the filesystem types on which no action should be taken. The prefix **no** has no effect when specified in an */etc/fstab* entry.

The prefix **no** can be meaningful with the **-a** option. For example, the command

mount -a -t nomsdos,smbfs

mounts all filesystems except those of type *msdos* and *smbfs*.

For most types all the **mount** program has to do is issue a simple mount(2) system call, and no detailed knowledge of the filesystem type is required. For a few types however (like nfs, nfs4, cifs, smbfs, ncpfs) an ad hoc code is necessary. The nfs, nfs4, cifs, smbfs, and ncpfs filesystems have a separate mount program. In order to make it possible to treat all types in a uniform way, **mount** will execute the program **/sbin/mount.***type* (if that exists) when called with type *type*. Since different versions of the **smbmount** program have different calling conventions, **/sbin/mount.smbfs** may have to

be a shell script that sets up the desired call.

-U, --uuid *uuid*
Mount the partition that has the specified *uuid*.

-v, --verbose
Verbose mode.

-w, --rw, --read-write
Mount the filesystem read/write. Read-write is the kernel default and the **mount** default is to try read-only if the previous mount(2) syscall with read-write flags on write-protected devices failed.

A synonym is **-o rw**.

Note that specifying **-w** on the command line forces **mount** to never try read-only mount on write-protected devices or already mounted read-only filesystems.

-V, --version
Display version information and exit.

-h, --help
Display help text and exit.

EXIT STATUS top

mount has the following exit status values (the bits can be ORed):

0
success

1
incorrect invocation or permissions

2
system error (out of memory, cannot fork, no more loop devices)

4
internal **mount** bug

8
user interrupt

16
problems writing or locking */etc/mtab*

32
mount failure

64
some mount succeeded

The command **mount -a** returns 0 (all succeeded), 32 (all failed), or 64 (some failed, some succeeded).

ENVIRONMENT top

LIBMOUNT_FSTAB=<path>
    overrides the default location of the *fstab* file (ignored for suid)

LIBMOUNT_MTAB=<path>
    overrides the default location of the *mtab* file (ignored for suid)

LIBMOUNT_DEBUG=all
    enables libmount debug output

LIBBLKID_DEBUG=all
    enables libblkid debug output

LOOPDEV_DEBUG=all
    enables loop device setup debug output

FILES top

See also **"The files /etc/fstab, /etc/mtab and /proc/mounts"** section above.

/etc/fstab
    filesystem table

/run/mount
    libmount private runtime directory

/etc/mtab
    table of mounted filesystems or symlink to */proc/mounts*

/etc/mtab~
    lock file (unused on systems with *mtab* symlink)

/etc/mtab.tmp
    temporary file (unused on systems with *mtab* symlink)

/etc/filesystems
    a list of filesystem types to try

HISTORY top

A **mount** command existed in Version 5 AT&T UNIX.

BUGS top

It is possible for a corrupted filesystem to cause a crash.

Some Linux filesystems don't support **-o sync** and **-o dirsync** (the ext2, ext3, ext4, fat and vfat filesystems *do* support synchronous updates (a la BSD) when mounted with the **sync** option).

The **-o remount** may not be able to change mount parameters (all *ext2fs*-specific parameters, except **sb**, are changeable with a remount, for example, but you can't change **gid** or **umask** for the *fatfs*).

It is possible that the files */etc/mtab* and */proc/mounts* don't match on systems with a regular *mtab* file. The first file is

based only on the **mount** command options, but the content of the second file also depends on the kernel and others settings (e.g. on a remote NFS server — in certain cases the **mount** command may report unreliable information about an NFS mount point and the */proc/mount* file usually contains more reliable information.) This is another reason to replace the *mtab* file with a symlink to the */proc/mounts* file.

Checking files on NFS filesystems referenced by file descriptors (i.e. the **fcntl** and **ioctl** families of functions) may lead to inconsistent results due to the lack of a consistency check in the kernel even if the **noac** mount option is used.

The **loop** option with the **offset** or **sizelimit** options used may fail when using older kernels if the **mount** command can't confirm that the size of the block device has been configured as requested. This situation can be worked around by using the losetup(8) command manually before calling **mount** with the configured loop device.

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