NAME

mount - mount a filesystem

SYNOPSIS

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

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

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

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

DESCRIPTION

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

mount -t*typ e device dir*

This tells the kernel to attach the filesystem found on *device* (which is of type *type*) at the directory *dir*. 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 ambivalent interpretation of the given argument. For example:

mount --target /mountpoint

The listing.

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.

The device indication.

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*. It is also possible to indicate a block special device using its filesystem label or UUID (see the **-L** and **-U** options below), or its partition label or UUID. (Partition identifiers are supported for example for GUID Partition Tables (GPT).)

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. **LABEL=***label*) rather than **/dev/disk/by-{label,uuid,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).**

Note that mount(8) 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.

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

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

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 simultaneously.

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** traditionally maintained a list of currently mounted filesystems in the file */etc/mtab*. This real mtab file is still supported, but on current Linux systems it is better to make it a symlink to */proc/mounts* instead, because a regular mtab file maintained in userspace cannot reliably work with namespaces, containers and other advanced Linux features.

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*. 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, PARTUUID or PARTLABEL) and *dir* are specified. For example, to mount device **foo** at **/dir**:

mount /dev/foo /dir

The 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 /dev/cdrom

or

mount /cd

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 member of the group of the special file.

The bind mounts.

Since Linux 2.4.0 it is possible to 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. One can also remount a single file (on a single file). It's also possible to use the bind mount to create a mount-point 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 is attached a second place by using:

mount --rbind *olddir newdir*

Note that the filesystem mount options will remain the same as those on the original mount point, and cannot be changed by passing the **-o** option along with **--bind**/**--rbind**. The mount options can be changed by a separate remount command, for example:

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

Note that the behavior of the remount operation depends on the /etc/mtab file. The first command stores the 'bind' flag in the /etc/mtab file and the second command reads the flag from the file. If you have a system without the /etc/mtab file or if you explicitly define source and target for the remount command (then **mount(8)** does not read /etc/mtab), then you have to use the bind flag (or option) for the remount command too. For example:

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

Note that **remount,ro,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.

The move operation.

Since Linux 2.5.1 it is possible to atomically move a **mounted tree** to another place. 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.

The 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.

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 to change multiple propagation flags with a single mount(2) syscall, and the flags cannot be mixed with other mount options.

Since util-linux 2.23 the **mount** command allows to use several propagation flags together and also together with other mount operations. This feature is EXPERIMENTAL. The propagation flags are applied by additional **mount(2)** syscalls 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

COMMAND-LINE OPTIONS

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 command **mount** 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*.

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

-c, **--no-canonicalize**
Don't canonicalize paths. The mount command canonicalizes all paths (from 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(8)** 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 go in parallel. A disadvantage is that the mounts are done in undefined order. 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 suid 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.

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

-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.

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

-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 target (mountpoint) or source (device). This option allows 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 target (mountpoint) or source (device). This option allows to explicitly define that the argument is the mount target.

-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(8)** 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** 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 blkid 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 option **-t** as well as in an */etc/fstab* entry. The list of filesystem types for option **-t** 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. This is the default. A synonym is **-o rw**.

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

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

FILESYSTEM-INDEPENDENT MOUNT OPTIONS

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 extN 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, fat, vfat and ufs):

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 **strictatime** and **relatime** 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).

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 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**.

Two other options are **fscontext=** and **defcontext=**, both of which are mutually exclusive of the context option. This means you can use fscontext and defcontext with each other, but neither can be used with context.

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(8) 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 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 file system.

diratime
Update directory inode access times on this filesystem. This is the default.

nodiratime
Do not update directory inode access times on this filesystem.

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

exec Permit execution of binaries.

noexec
Do not permit direct execution of any binaries on the mounted filesystem. (Until recently it was possible to run binaries anyway using a command like /lib/ld*.so /mnt/binary. This trick fails since Linux 2.4.25 / 2.6.0.)

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** 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 i-node was written to disk.

nolazytime
Do not use the nolazytime feature.

suid Allow set-user-identifier or set-group-identifier bits to take effect.

nosuid
Do not allow set-user-identifier or set-group-identifier bits to take effect.

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 functionality follows the standard way the mount command works with options from fstab. This means that the mount command only doesn't read fstab (or mtab) when both the *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 found in fstab than remount with unspecified source is allowed.

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 the mtab file, nor sent to the mount.*type* helpers nor to the mount(2) system call. The suggested format is **x-***appname.option* (e.g. **x-systemd.automount**).

x-mount.mkdir[=*mode*]
Allow to make a target directory (mountpoint). 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.

FILESYSTEM-SPECIFIC MOUNT OPTIONS

The following options apply only to certain filesystems. We sort them by filesystem. They all follow the **-o** flag.

What options are supported depends a bit on the running kernel. More info may be found in the kernel source subdirectory *Documentation/filesystems*.

Mount options for adfs

uid=*value* and **gid**=*value*
Set the owner and group of the files in the filesystem (default: uid=gid=0).

ownmask=*value* and **othmask**=*value*
Set the permission mask for ADFS 'owner' permissions and 'other' permissions, respectively (default: 0700 and 0077, respectively). See also*/usr/sr c/linux/Documentation/filesystems/adfs.txt*.

Mount options for affs

uid=*value* and **gid**=*value*
Set the owner and group of the root of the filesystem (default: uid=gid=0, but with option **uid** or **gid** without specified value, the uid and gid of the current process are taken).

setuid=*value* and **setgid**=*value*
Set the owner and group of all files.

mode=*value*
Set the mode of all files to *value* & 0777 disregarding the original permissions. Add search permission to directories that have read permission. The value is given in octal.

protect
Do not allow any changes to the protection bits on the filesystem.

usemp
Set uid and gid of the root of the filesystem to the uid and gid of the mount point upon the first sync or umount, and then clear this option. Strange...

verbose
Print an informational message for each successful mount.

prefix=*string*
Prefix used before volume name, when following a link.

volume=*string*
Prefix (of length at most 30) used before '/' when following a symbolic link.

reserved=*value*
(Default: 2.) Number of unused blocks at the start of the device.

root=*value*
Give explicitly the location of the root block.

bs=*value*
Give blocksize. Allowed values are 512, 1024, 2048, 4096.

grpquota|noquota|quota|usrquota
These options are accepted but ignored. (However, quota utilities may react to such strings in */etc/fstab*.)

Mount options for btrfs

Btrfs is a copy-on-write filesystem for Linux aimed at implementing advanced features while focusing on fault tolerance, repair, and easy administration.

alloc_start=*bytes*
Debugging option to force all block allocations above a certain byte threshold on each block device. The value is specified in bytes, optionally with a K, M, or G suffix, case insensitive. Default is 1MB.

autodefrag
Disable/enable auto defragmentation. Auto defragmentation detects small random writes into files and queues them up for the defrag process. Works best for small files; not well-suited for large database workloads.

check_int|check_int_data|check_int_print_mask=*value*
These debugging options control the behavior of the integrity checking module(the BTRFS_FS_CHECK_INTEGRITY config option required).

check_int enables the integrity checker module, which examines all block-write requests to ensure on-disk consistency, at a large memory and CPU cost.

check_int_data includes extent data in the integrity checks, and implies the check_int option.

check_int_print_mask takes a bitmask of BTRFSIC_PRINT_MASK_* values as defined in fs/btrfs/check-integrity.c, to control the integrity checker module behavior.

See comments at the top of *fs/btrfs/check-integrity.c* for more info.

commit=*seconds*
Set the interval of periodic commit, 30 seconds by default. Higher values defer data being synced to permanent storage, with obvious consequences when the system crashes. The upper bound is not forced, but a warning is printed if it's more than 300 seconds (5 minutes).

compress|compress=*type*|**compress-force|compress-force**=*type*
Control BTRFS file data compression. Type may be specified as zlib lzo or no (for no compression, used for remounting). If no type is specified, zlib is used. If **compress-force** is specified, all files will be compressed, whether or not they compress well. If compression is enabled, **nodatacow** and **nodatasum** are disabled.

degraded
Allow mounts to continue with missing devices. A read-write mount may fail with too many devices missing, for example if a stripe member is completely missing.

device=*devicepath*
Specify a device during mount so that ioctls on the control device can be avoided. Especially useful when trying to mount a multi-device setup as root. May be specified multiple times for multiple devices.

discard
Disable/enable the discard mount option. The discard function issues frequent commands to let the block device reclaim space freed by the filesystem. This is useful for SSD devices, thinly provisioned LUNs and virtual machine images, but may have a significant performance impact. (The **fstrim** command is also available to initiate batch trims from userspace.)

enospc_debug
Disable/enable debugging option to be more verbose in some ENOSPC conditions.

fatal_errors=*action*
Action to take when encountering a fatal error: bug - BUG() on a fatal error. This is the default. panic - panic() on a fatal error.

flushoncommit
The **flushoncommit** mount option forces any data dirtied by a write in a prior transaction to commit as part of the current commit. This makes the committed state a fully consistent view of the filesystem from the application's perspective (i.e., it includes all completed filesystem operations). This was previously the behavior only when a snapshot is created.

inode_cache
Enable free inode number caching. Defaults to off due to an overflow problem when the free space CRCs don't fit inside a single page.

max_inline=*bytes*
Specify the maximum amount of space, in bytes, that can be inlined in a metadata B-tree leaf. The value is specified in bytes, optionally with a K, M, or G suffix, case insensitive. In practice, this value is limited by the root sector size, with some space unavailable due to leaf headers. For a 4k sectorsize, max inline data is ~3900 bytes.

metadata_ratio=*value*
Specify that 1 metadata chunk should be allocated after every *value* data chunks. Off by default.

noacl Enable/disable support for Posix Access Control Lists (ACLs). See the acl(5) manual page for more information about ACLs.

nobarrier
Enable/disable the use of block-layer write barriers. Write barriers ensure that certain IOs make it through the device cache and are on persistent storage. If disabled on a device with a volatile (non-battery-backed) write-back cache, the **nobarrier** option will lead to filesystem corruption on a system crash or power loss.

nodatacow
Enable/disable data copy-on-write for newly created files. This option implies **nodatasum**, and disables all compression.

nodatasum
Enable/disable data checksumming for newly created files. This option implies **datacow**.

notreelog
Enable/disable the tree logging used for fsync and O_SYNC writes.

recovery
Enable autorecovery attempts if a bad tree root is found at mount time. Currently this scans a list of several previous tree roots and tries to use the first readable.

rescan_uuid_tree
Force check and rebuild procedure of the UUID tree. This should not normally be needed.

skip_balance
Skip automatic resume of an interrupted balance operation after mount. May be resumed with btrfs balance resume.

nospace_cache
Disable freespace cache loading without clearing the cache.

clear_cache
Force clearing and rebuilding of the disk space cache if something has gone wrong.

ssd|nossd|ssd_spread
Options to control ssd allocation schemes. By default, BTRFS will enable or disable ssd allocation heuristics depending on whether a rotational or nonrotational disk is in use. The **ssd** and **nossd** options can override this autodetection.

The **ssd_spread** mount option attempts to allocate into big chunks of unused space, and may perform better on low-end ssds. **ssd_spread** implies **ssd**, enabling all other ssd heuristics as well.

subvol=*path*
Mount subvolume at *path* rather than the root subvolume. The *path* is relative to the top level subvolume.

subvolid=*ID*
Mount subvolume specified by an ID number rather than the root subvolume. This allows mounting of subvolumes which are not in the root of the mounted filesystem. You can use btrfs subvolume list to see subvolume ID numbers.

subvolrootid=*objectid* (deprecated)
Mount subvolume specified by *objectid* rather than the root subvolume. This allows mounting of subvolumes which are not in the root of the mounted filesystem. You can use btrfs subvolume show to see the object ID for a subvolume.

thread_pool=*number*
The number of worker threads to allocate. The default number is equal to the number of CPUs + 2, or 8, whichever is smaller.

user_subvol_rm_allowed
Allow subvolumes to be deleted by a non-root user. Use with caution.

Mount options for cifs

See the options section of the **mount.cifs(8)** man page (cifs-utils package must be installed).

Mount options for coherent

None.

Mount options for debugfs

The debugfs filesystem is a pseudo filesystem, traditionally mounted on */sys/kernel/debug*. As of kernel version 3.4, debugfs has the following options:

uid=*n*, **gid=***n*
Set the owner and group of the mountpoint.

mode=*value*
Sets the mode of the mountpoint.

Mount options for devpts

The devpts filesystem is a pseudo filesystem, traditionally mounted on */dev/pts*. In order to acquire a pseudo terminal, a process opens */dev/ptmx*; the number of the pseudo terminal is then

made available to the process and the pseudo terminal slave can be accessed as */dev/pts/*<number>.

uid=*value* and **gid**=*value*
This sets the owner or the group of newly created PTYs to the specified values. When nothing is specified, they will be set to the UID and GID of the creating process. For example, if there is a tty group with GID 5, then **gid=5** will cause newly created PTYs to belong to the tty group.

mode=*value*
Set the mode of newly created PTYs to the specified value. The default is 0600. A value of **mode=620** and **gid=5** makes mesg y the default on newly created PTYs.

newinstance
Create a private instance of devpts filesystem, such that indices of ptys allocated in this new instance are independent of indices created in other instances of devpts.

All mounts of devpts without this **newinstance** option share the same set of pty indices (i.e legacy mode). Each mount of devpts with the **newinstance** option has a private set of pty indices.

This option is mainly used to support containers in the linux kernel. It is implemented in linux kernel versions starting with 2.6.29. Further, this mount option is valid only if CONFIG_DEVPTS_MULTIPLE_INSTANCES is enabled in the kernel configuration.

To use this option effectively, */dev/ptmx* must be a symbolic link to *pts/ptmx*. See *Documentation/filesystems/devpts.txt* in the linux kernel source tree for details.

ptmxmode=*value*

Set the mode for the new *ptmx* device node in the devpts filesystem.

With the support for multiple instances of devpts (see **newinstance** option above), each instance has a private *ptmx* node in the root of the devpts filesystem (typically */dev/pts/ptmx*).

For compatibility with older versions of the kernel, the default mode of the new *ptmx* node is 0000. **ptmxmode**=*value* specifies a more useful mode for the *ptmx* node and is highly recommended when the **newinstance** option is specified.

This option is only implemented in linux kernel versions starting with 2.6.29. Further, this option is valid only if CONFIG_DEVPTS_MULTIPLE_INSTANCES is enabled in the kernel configuration.

Mount options for ext2

The 'ext2' filesystem is the standard Linux filesystem. Since Linux 2.5.46, for most mount options the default is determined by the filesystem superblock. Set them with tune2fs(8).

acl|noacl
Support POSIX Access Control Lists (or not).

bsddf|minixdf
Set the behavior for the *statfs* system call. The **minixdf** behavior is to return in the *f_blocks* field the total number of blocks of the filesystem, while the **bsddf** behavior (which is the default) is to subtract the overhead blocks used by the ext2 filesystem and not available for file storage. Thus

% mount /k -o minixdf; df /k; umount /k

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2630655	86954	2412169	3%	/k

% mount /k -o bsddf; df /k; umount /k

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2543714	13	2412169	0%	/k

(Note that this example shows that one can add command-line options to the options given in */etc/fstab*.)

check=none or **nocheck**
No checking is done at mount time. This is the default. This is fast. It is wise to invoke e2fsck(8) every now and then, e.g. at boot time. The non-default behavior is unsupported (check=normal and check=strict options have been removed). Note that these mount options don't have to be supported if ext4 kernel driver is used for ext2 and ext3 filesystems.

debug Print debugging info upon each (re)mount.

errors={continue|remount-ro|panic}
Define the behavior when an error is encountered. (Either ignore errors and just mark the filesystem erroneous and continue, or remount the filesystem read-only, or panic and halt the system.) The default is set in the filesystem superblock, and can be changed using tune2fs(8).

grpid|bsdgroups and **nogrpid|sysvgroups**
These options define what group id a newly created file gets. When **grpid** is set, it takes the group id of the directory in which it is created; otherwise (the default) it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

grpquota|noquota|quota|usrquota
The usrquota (same as quota) mount option enables user quota support on the filesystem. grpquota enables group quotas support. You need the quota utilities to actually enable and manage the quota system.

nouid32
Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values.

oldalloc or **orlov**
Use old allocator or Orlov allocator for new inodes. Orlov is default.

resgid=*n* and **resuid=***n*
The ext2 filesystem reserves a certain percentage of the available space (by default 5%, see mke2fs(8) and tune2fs(8)). These options determine who can use the reserved blocks. (Roughly: whoever has the specified uid, or belongs to the specified group.)

sb=*n* Instead of block 1, use block *n* as superblock. This could be useful when the filesystem has been damaged. (Earlier, copies of the superblock would be made every 8192 blocks: in block 1, 8193, 16385, ... (and one got thousands of copies on a big filesystem). Since version 1.08, **mke2fs** has a -s (sparse superblock) option to reduce the number of backup superblocks, and since version 1.15 this is the default. Note that this may mean that ext2 filesystems created by a recent **mke2fs** cannot be mounted r/w under Linux 2.0.*.) The block number here uses 1 k units. Thus, if you want to use logical block 32768 on a filesystem with 4 k blocks, use sb=131072.

user_xattr|nouser_xattr
Support user. extended attributes (or not).

Mount options for ext3

The ext3 filesystem is a version of the ext2 filesystem which has been enhanced with journaling. It supports the same options as ext2 as well as the following additions:

journal=update
Update the ext3 filesystem's journal to the current format.

journal=inum
When a journal already exists, this option is ignored. Otherwise, it specifies the number of the inode which will represent the ext3 filesystem's journal file; ext3 will create a new journal, overwriting the old contents of the file whose inode number is *inum*.

journal_dev=devnum/journal_path=path
When the external journal device's major/minor numbers have changed, these options allow the user to specify the new journal location. The journal device is identified either through its new major/minor numbers encoded in devnum, or via a path to the device.

norecovery/noload
Don't load the journal on mounting. Note that if the filesystem was not unmounted cleanly, skipping the journal replay will lead to the filesystem containing inconsistencies that can lead to any number of problems.

data={journal|ordered|writeback}
Specifies the journaling mode for file data. Metadata is always journaled. To use modes other than **ordered** on the root filesystem, pass the mode to the kernel as boot parameter, e.g. *rootflags=data=journal*.

journal
All data is committed into the journal prior to being written into the main filesystem.

ordered
This is the default mode. All data is forced directly out to the main file system prior to its metadata being committed to the journal.

writeback
Data ordering is not preserved – data may be written into the main filesystem after its metadata has been committed to the journal. This is rumoured to be the highest-throughput option. It guarantees internal filesystem integrity, however it can allow old data to appear in files after a crash and journal recovery.

data_err=ignore
Just print an error message if an error occurs in a file data buffer in ordered mode.

data_err=abort
Abort the journal if an error occurs in a file data buffer in ordered mode.

barrier=0 / barrier=1
This disables / enables the use of write barriers in the jbd code. barrier=0 disables, barrier=1 enables (default). This also requires an IO stack which can support barriers, and if jbd gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

commit=*nrsec*
Sync all data and metadata every *nrsec* seconds. The default value is 5 seconds. Zero means default.

user_xattr
Enable Extended User Attributes. See the **attr(5)** manual page.

acl Enable POSIX Access Control Lists. See the acl(5) manual page.

usrjquota=aquota.user|grpjquota=aquota.group|jqfmt=vfsv0
Apart from the old quota system (as in ext2, jqfmt=vfsold aka version 1 quota) ext3 also supports journaled quotas (version 2 quota). jqfmt=vfsv0 enables journaled quotas. For journaled quotas the mount options usrjquota=aquota.user and grpjquota=aquota.group are required to tell the quota system which quota database files to use. Journaled quotas have the advantage that even after a crash no quota check is required.

Mount options for ext4

The ext4 filesystem is an advanced level of the ext3 filesystem which incorporates scalability and reliability enhancements for supporting large filesystem.

The options **journal_dev, norecovery, noload, data, commit, orlov, oldalloc, [no]user_xattr [no]acl, bsddf, minixdf, debug, errors, data_err, grpid, bsdgroups, nogrpid sysvgroups, resgid, resuid, sb, quota, noquota, grpquota, usrquota usrjquota, grpjquota and jqfmt** are backwardly compatible with ext3 or ext2.

journal_checksum
Enable checksumming of the journal transactions. This will allow the recovery code in e2fsck and the kernel to detect corruption in the kernel. It is a compatible change and will be ignored by older kernels.

journal_async_commit
Commit block can be written to disk without waiting for descriptor blocks. If enabled, older kernels cannot mount the device. This will enable 'journal_checksum' internally.

barrier=0 / barrier=1 / barrier / nobarrier
These mount options have the same effect as in ext3. The mount options barrier and nobarrier are added for consistency with other ext4 mount options.

The ext4 filesystem enables write barriers by default.

inode_readahead_blks=*n*
This tuning parameter controls the maximum number of inode table blocks that ext4's inode table readahead algorithm will pre-read into the buffer cache. The value must be a power of 2. The default value is 32 blocks.

stripe=*n*
Number of filesystem blocks that mballoc will try to use for allocation size and alignment. For RAID5/6 systems this should be the number of data disks * RAID chunk size in filesystem blocks.

delalloc
Deferring block allocation until write-out time.

nodelalloc
Disable delayed allocation. Blocks are allocated when data is copied from user to page cache.

max_batch_time=*usec*
Maximum amount of time ext4 should wait for additional filesystem operations to be batch together with a synchronous write operation. Since a synchronous write operation is going to force a commit and then a wait for the I/O complete, it doesn't cost much, and can be a huge throughput win, we wait for a small amount of time to see if any other transactions can piggyback on the synchronous write. The algorithm used is designed to automatically tune for the speed of the disk, by measuring the amount of time (on average) that it takes to finish committing a transaction. Call this time the commit time. If the time that the transaction has been running is less than the commit time, ext4 will try sleeping for the commit time to see if other operations will join the transaction. The commit time is capped by the max_batch_time, which defaults to 15000µs (15 ms). This

optimization can be turned off entirely by setting max_batch_time to 0.

min_batch_time=*usec*
This parameter sets the commit time (as described above) to be at least min_batch_time. It defaults to zero microseconds. Increasing this parameter may improve the throughput of multi-threaded, synchronous workloads on very fast disks, at the cost of increasing latency.

journal_ioprio=*prio*
The I/O priority (from 0 to 7, where 0 is the highest priority) which should be used for I/O operations submitted by kjournald2 during a commit operation. This defaults to 3, which is a slightly higher priority than the default I/O priority.

abort Simulate the effects of calling ext4_abort() for debugging purposes. This is normally used while remounting a filesystem which is already mounted.

auto_da_alloc|noauto_da_alloc
Many broken applications don't use fsync() when replacing existing files via patterns such as

fd = open(foo.new)/write(fd,...)/close(fd)/ rename(foo.new, foo)

or worse yet

fd = open(foo, O_TRUNC)/write(fd,...)/close(fd).

If auto_da_alloc is enabled, ext4 will detect the replace-via-rename and replace-via-truncate patterns and force that any delayed allocation blocks are allocated such that at the next journal commit, in the default data=ordered mode, the data blocks of the new file are forced to disk before the rename() operation is committed. This provides roughly the same level of guarantees as ext3, and avoids the zero-length problem that can happen when a system crashes before the delayed allocation blocks are forced to disk.

noinit_itable
Do not initialize any uninitialized inode table blocks in the background. This feature may be used by installation CD's so that the install process can complete as quickly as possible; the inode table initialization process would then be deferred until the next time the filesystem is mounted.

init_itable=n
The lazy itable init code will wait n times the number of milliseconds it took to zero out the previous block group's inode table. This minimizes the impact on system performance while the filesystem's inode table is being initialized.

discard/nodiscard
Controls whether ext4 should issue discard/TRIM commands to the underlying block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs, but it is off by default until sufficient testing has been done.

nouid32
Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values.

block_validity/noblock_validity
This options allows to enables/disables the in-kernel facility for tracking filesystem metadata blocks within internal data structures. This allows multi-block allocator and other routines to quickly locate extents which might overlap with filesystem metadata blocks. This option is intended for debugging purposes and since it negatively affects the performance, it is off by default.

dioread_lock/dioread_nolock
Controls whether or not ext4 should use the DIO read locking. If the dioread_nolock option is specified ext4 will allocate uninitialized extent before buffer write and convert the extent to initialized after IO completes. This approach allows ext4 code to avoid using inode mutex, which improves scalability on high speed storages. However this does not work with data journaling and dioread_nolock option will be ignored with kernel warning. Note that dioread_nolock code path is only used for extent-based files. Because of the restrictions this options comprises it is off by default (e.g. dioread_lock).

max_dir_size_kb=n
This limits the size of the directories so that any attempt to expand them beyond the specified limit in kilobytes will cause an ENOSPC error. This is useful in memory-constrained environments, where a very large directory can cause severe performance problems or even provoke the Out Of Memory killer. (For example, if there is only 512 MB memory available, a 176 MB directory may seriously cramp the system's style.)

i_version
Enable 64-bit inode version support. This option is off by default.

Mount options for fat

(Note: *fat* is not a separate filesystem, but a common part of the *msdos*, *umsdos* and *vfat* filesystems.)

blocksize={512|1024|2048}
Set blocksize (default 512). This option is obsolete.

uid=*value* and **gid=***value*
Set the owner and group of all files. (Default: the uid and gid of the current process.)

umask=*value*
Set the umask (the bitmask of the permissions that are **not** present). The default is the umask of the current process. The value is given in octal.

dmask=*value*
Set the umask applied to directories only. The default is the umask of the current process. The value is given in octal.

fmask=*value*
Set the umask applied to regular files only. The default is the umask of the current process. The value is given in octal.

allow_utime=*value*
This option controls the permission check of mtime/atime.

20 If current process is in group of file's group ID, you can change timestamp.

2 Other users can change timestamp.

The default is set from 'dmask' option. (If the directory is writable, utime(2) is also allowed. I.e. ~dmask & 022)

Normally utime(2) checks current process is owner of the file, or it has CAP_FOWNER capability. But FAT filesystem doesn't have uid/gid on disk, so normal check is too inflexible. With this option you can relax it.

check=*value*
Three different levels of pickiness can be chosen:

r[**elaxed**]
Upper and lower case are accepted and equivalent, long name parts are truncated (e.g. *verylongname.foobar* becomes *verylong.foo*), leading and embedded spaces are accepted in each name part (name and extension).

n[**ormal**]
Like relaxed, but many special characters (*, ?, <, spaces, etc.) are rejected. This is the default.

s[**trict**]
Like normal, but names that contain long parts or special characters that are sometimes used on Linux but are not accepted by MS-DOS (+, =, etc.) are rejected.

codepage=*value*
Sets the codepage for converting to shortname characters on FAT and VFAT filesystems. By default, codepage 437 is used.

conv=*mode*
The *fat* filesystem can perform CRLF<-->NL conversion (MS-DOS text format to UNIX text format) in the kernel. The following conversion *mode*s are available:

b[**inary**]
No translation is performed. This is the default.

t[**ext**] CRLF<-->NL translation is performed on all files.

a[**uto**] CRLF<-->NL translation is performed on all files that don't have a well-known binary extension. The list of known extensions can be found at the beginning of *fs/fat/misc.c* (as of 2.0, the list is: exe, com, bin, app, sys, drv, ovl, ovr, obj, lib, dll, pif, arc, zip, lha, lzh, zoo, tar, z, arj, tz, taz, tzp, tpz, gz, tgz, deb, gif, bmp, tif, gl, jpg, pcx, tfm, vf, gf, pk, pxl, dvi).

Programs that do computed lseeks won't like in-kernel text conversion. Several people have had their data ruined by this translation. Beware!

For filesystems mounted in binary mode, a conversion tool (fromdos/todos) is available. This option is obsolete.

cvf_format=*module*
Forces the driver to use the CVF (Compressed Volume File) module cvf_*module* instead of auto-detection. If the kernel supports kmod, the cvf_format=xxx option also controls on-demand CVF module loading. This option is obsolete.

cvf_option=*option*
Option passed to the CVF module. This option is obsolete.

debug Turn on the *debug* flag. A version string and a list of filesystem parameters will be printed (these data are also printed if the parameters appear to be inconsistent).

discard
If set, causes discard/TRIM commands to be issued to the block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs.

dos1xfloppy
If set, use a fallback default BIOS Parameter Block configuration, determined by backing device size. These static parameters match defaults assumed by DOS 1.x for 160 kiB, 180 kiB, 320 kiB, and 360 kiB floppies and floppy images.

errors={panic|continue|remount-ro}
Specify FAT behavior on critical errors: panic, continue without doing anything, or remount the partition in read-only mode (default behavior).

fat={12|16|32}
Specify a 12, 16 or 32 bit fat. This overrides the automatic FAT type detection routine. Use with caution!

iocharset=*value*
Character set to use for converting between 8 bit characters and 16 bit Unicode characters. The default is iso8859-1. Long filenames are stored on disk in Unicode format.

nfs={stale_rw|nostale_ro}
Enable this only if you want to export the FAT filesystem over NFS.

stale_rw: This option maintains an index (cache) of directory inodes which is used by the nfs-related code to improve look-ups. Full file operations (read/write) over NFS are supported but with cache eviction at NFS server, this could result in spurious **ESTALE** errors.

nostale_ro: This option bases the inode number and filehandle on the on-disk location of a file in the FAT directory entry. This ensures that**EST ALE** will not be returned after a file is evicted from the inode cache. However, it means that operations such as rename, create and unlink could cause filehandles that previously pointed at one file to point at a different file, potentially causing data corruption. For this reason, this option also mounts the filesystem readonly.

To maintain backward compatibility, '-o nfs' is also accepted, defaulting to **stale_rw**.

tz=UTC
This option disables the conversion of timestamps between local time (as used by Windows on FAT) and UTC (which Linux uses internally). This is particularly useful when mounting devices (like digital cameras) that are set to UTC in order to avoid the pitfalls of local time.

time_offset=*minutes*
Set offset for conversion of timestamps from local time used by FAT to UTC. I.e., *minutes* minutes will be subtracted from each timestamp to convert it to UTC used internally by Linux. This is useful when the time zone set in the kernel via settimeofday(2) is not the time zone used by the filesystem. Note that this option still does not provide correct time stamps in all cases in presence of DST - time stamps in a different DST setting will be off by one hour.

quiet Turn on the *quiet* flag. Attempts to chown or chmod files do not return errors, although they fail. Use with caution!

rodir FAT has the ATTR_RO (read-only) attribute. On Windows, the ATTR_RO of the directory will just be ignored, and is used only by applications as a flag (e.g. it's set for the customized folder).

If you want to use ATTR_RO as read-only flag even for the directory, set this option.

showexec
If set, the execute permission bits of the file will be allowed only if the extension part of the name is .EXE, .COM, or .BAT. Not set by default.

sys_immutable
If set, ATTR_SYS attribute on FAT is handled as IMMUTABLE flag on Linux. Not set by default.

flush If set, the filesystem will try to flush to disk more early than normal. Not set by default.

usefree
Use the free clusters value stored on FSINFO. It'll be used to determine number of free clusters without scanning disk. But it's not used by default, because recent Windows don't update it correctly in some case. If you are sure the free clusters on FSINFO is correct, by this option you can avoid scanning disk.

dots, **nodots**, **dotsOK=**[**yes**|**no**]
Various misguided attempts to force Unix or DOS conventions onto a FAT filesystem.

Mount options for hfs

creator=*cccc*, **type=***cccc*
Set the creator/type values as shown by the MacOS finder used for creating new files. Default values: '????'.

uid=*n*, **gid=***n*
Set the owner and group of all files. (Default: the uid and gid of the current process.)

dir_umask=*n*, **file_umask=***n*, **umask=***n*
Set the umask used for all directories, all regular files, or all files and directories. Defaults to the umask of the current process.

session=*n*
Select the CDROM session to mount. Defaults to leaving that decision to the CDROM driver. This option will fail with anything but a CDROM as underlying device.

part=*n*
Select partition number n from the device. Only makes sense for CDROMs. Defaults to not parsing the partition table at all.

quiet Don't complain about invalid mount options.

Mount options for hpfs

uid=*value* and **gid=***value*
Set the owner and group of all files. (Default: the uid and gid of the current process.)

umask=*value*
Set the umask (the bitmask of the permissions that are **not** present). The default is the umask of the current process. The value is given in octal.

case={**lower**|**asis**}
Convert all files names to lower case, or leave them. (Default: **case=lower**.)

conv={**binary**|**text**|**auto**}
For **conv=text**, delete some random CRs (in particular, all followed by NL) when reading a file. For **conv=auto**, choose more or less at random between **conv=binary** and **conv=text**. For **conv=binary**, just read what is in the file. This is the default.

nocheck
Do not abort mounting when certain consistency checks fail.

Mount options for iso9660

ISO 9660 is a standard describing a filesystem structure to be used on CD-ROMs. (This filesystem type is also seen on some DVDs. See also the *udf* filesystem.)

Normal *iso9660* filenames appear in a 8.3 format (i.e., DOS-like restrictions on filename length), and in addition all characters are in upper case. Also there is no field for file ownership, protection, number of links, provision for block/character devices, etc.

Rock Ridge is an extension to iso9660 that provides all of these UNIX-like features. Basically there are extensions to each directory record that supply all of the additional information, and when Rock Ridge is in use, the filesystem is indistinguishable from a normal UNIX filesystem (except that it is read-only, of course).

norock
Disable the use of Rock Ridge extensions, even if available. Cf. **map**.

nojoliet
Disable the use of Microsoft Joliet extensions, even if available. Cf. **map**.

check={r[elaxed]|s[trict]}
With **check=relaxed**, a filename is first converted to lower case before doing the lookup. This is probably only meaningful together with **norock** and **map=normal**. (Default: **check=strict**.)

uid=*value* and **gid=***value*
Give all files in the filesystem the indicated user or group id, possibly overriding the information found in the Rock Ridge extensions. (Default: **uid=0,gid=0**.)

map={n[ormal]|o[ff]|a[corn]}
For non-Rock Ridge volumes, normal name translation maps upper to lower case ASCII, drops a trailing ';1', and converts ';' to '.'. With **map=off** no name translation is done. See **norock**. (Default:**map=normal**.) **map=acorn** is lik e **map=normal** but also apply Acorn extensions if present.

mode=*value*
For non-Rock Ridge volumes, give all files the indicated mode. (Default: read and execute permission for everybody.) Since Linux 2.1.37 one no longer needs to specify the mode in decimal. (Octal is indicated by a leading 0.)

unhide
Also show hidden and associated files. (If the ordinary files and the associated or hidden files have the same filenames, this may make the ordinary files inaccessible.)

block={512|1024|2048}
Set the block size to the indicated value. (Default:**bloc k=1024**.)

conv={a[uto]|b[inary]|m[text]|t[ext]}
(Default: **conv=binary**.) Since Linux 1.3.54 this option has no effect anymore. (And non-binary settings used to be very dangerous, possibly leading to silent data corruption.)

cruft If the high byte of the file length contains other garbage, set this mount option to ignore the high order bits of the file length. This implies that a file cannot be larger than 16 MB.

session=*x*
Select number of session on multisession CD. (Since 2.3.4.)

sbsector=*xxx*
Session begins from sector xxx. (Since 2.3.4.)

The following options are the same as for vfat and specifying them only makes sense when using discs encoded using Microsoft's Joliet extensions.

iocharset=*value*
Character set to use for converting 16 bit Unicode characters on CD to 8 bit characters. The default is iso8859-1.

utf8 Convert 16 bit Unicode characters on CD to UTF-8.

Mount options for jfs

iocharset=*name*
Character set to use for converting from Unicode to ASCII. The default is to do no conversion. Use **iocharset=utf8** for UTF8 translations. This requires CONFIG_NLS_UTF8 to be set in the kernel *.config* file.

resize=*value*
Resize the volume to *value* blocks. JFS only supports growing a volume, not shrinking it. This option is only valid during a remount, when the volume is mounted read-write. The **resize** keyword with no value will grow the volume to the full size of the partition.

nointegrity
Do not write to the journal. The primary use of this option is to allow for higher performance when restoring a volume from backup media. The integrity of the volume is not guaranteed if the system abnormally ends.

integrity
Default. Commit metadata changes to the journal. Use this option to remount a volume where the **nointegrity** option was previously specified in order to restore normal behavior.

errors={continue|remount-ro|panic}
Define the behavior when an error is encountered. (Either ignore errors and just mark the filesystem erroneous and continue, or remount the filesystem read-only, or panic and halt the system.)

noquota|quota|usrquota|grpquota
These options are accepted but ignored.

Mount options for minix
None.

Mount options for msdos
See mount options for fat. If the *msdos* filesystem detects an inconsistency, it reports an error and sets the file system read-only. The filesystem can be made writable again by remounting it.

Mount options for ncpfs
Just like *nfs*, the *ncpfs* implementation expects a binary argument (a *struct ncp_mount_data*) to the mount system call. This argument is constructed by **ncpmount(8)** and the current version of **mount** (2.12) does not know anything about ncpfs.

Mount options for nfs and nfs4
See the options section of the nfs(5) man page (nfs-utils package must be installed).

The *nfs* and *nfs4* implementation expects a binary argument (a *struct nfs_mount_data*) to the mount system call. This argument is constructed by mount.nfs(8) and the current version of **mount** (2.13) does not know anything about nfs and nfs4.

Mount options for ntfs
iocharset=*name*
Character set to use when returning file names. Unlike VFAT, NTFS suppresses names that contain nonconvertible characters. Deprecated.

nls=*name*
New name for the option earlier called *iocharset*.

utf8 Use UTF-8 for converting file names.

uni_xlate={0|1|2}
For 0 (or 'no' or 'false'), do not use escape sequences for unknown Unicode characters. For 1 (or 'yes' or 'true') or 2, use vfat-style 4-byte escape sequences starting with :. Here 2 give a little-endian encoding and 1 a byteswapped bigendian encoding.

posix=[0|1]
If enabled (posix=1), the filesystem distinguishes between upper and lower case. The 8.3 alias names are presented as hard links instead of being suppressed. This option is obsolete.

uid=*value*, **gid=***value* and **umask=***value*
Set the file permission on the filesystem. The umask value is given in octal. By default, the files are owned by root and not readable by somebody else.

Mount options for proc

uid=*value* and **gid=***value*

These options are recognized, but have no effect as far as I can see.

Mount options for ramfs

Ramfs is a memory based filesystem. Mount it and you have it. Unmount it and it is gone. Present since Linux 2.3.99pre4. There are no mount options.

Mount options for reiserfs

Reiserfs is a journaling filesystem.

conv Instructs version 3.6 reiserfs software to mount a version 3.5 filesystem, using the 3.6 format for newly created objects. This filesystem will no longer be compatible with reiserfs 3.5 tools.

hash={rupasov|tea|r5|detect}

Choose which hash function reiserfs will use to find files within directories.

rupasov

A hash invented by Yury Yu. Rupasov. It is fast and preserves locality, mapping lexicographically close file names to close hash values. This option should not be used, as it causes a high probability of hash collisions.

tea A Davis-Meyer function implemented by Jeremy Fitzhardinge. It uses hash permuting bits in the name. It gets high randomness and, therefore, low probability of hash collisions at some CPU cost. This may be used if EHASHCOLLISION errors are experienced with the r5 hash.

r5 A modified version of the rupasov hash. It is used by default and is the best choice unless the filesystem has huge directories and unusual file-name patterns.

detect Instructs *mount* to detect which hash function is in use by examining the filesystem being mounted, and to write this information into the reiserfs superblock. This is only useful on the first mount of an old format filesystem.

hashed_relocation

Tunes the block allocator. This may provide performance improvements in some situations.

no_unhashed_relocation

Tunes the block allocator. This may provide performance improvements in some situations.

noborder

Disable the border allocator algorithm invented by Yury Yu. Rupasov. This may provide performance improvements in some situations.

nolog Disable journaling. This will provide slight performance improvements in some situations at the cost of losing reiserfs's fast recovery from crashes. Even with this option turned on, reiserfs still performs all journaling operations, save for actual writes into its journaling area. Implementation of *nolog* is a work in progress.

notail By default, reiserfs stores small files and 'file tails' directly into its tree. This confuses some utilities such as **LILO(8).** This option is used to disable packing of files into the tree.

replayonly

Replay the transactions which are in the journal, but do not actually mount the filesystem. Mainly used by *reiserfsck*.

resize=*number*
A remount option which permits online expansion of reiserfs partitions. Instructs reiserfs to assume that the device has *number* blocks. This option is designed for use with devices which are under logical volume management (LVM). There is a special*r esizer* utility which can be obtained from *ftp://ftp.namesys.com/pub/reiserfsprogs*.

user_xattr
Enable Extended User Attributes. See the **attr(5)** manual page.

acl Enable POSIX Access Control Lists. See the acl(5) manual page.

barrier=none / barrier=flush
This disables / enables the use of write barriers in the journaling code. barrier=none disables, barrier=flush enables (default). This also requires an IO stack which can support barriers, and if reiserfs gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

Mount options for romfs
None.

Mount options for squashfs
None.

Mount options for smbfs
Just like *nfs*, the *smbfs* implementation expects a binary argument (a *struct smb_mount_data*) to the mount system call. This argument is constructed by **smbmount(8)** and the current version of **mount** (2.12) does not know anything about smbfs.

Mount options for sysv
None.

Mount options for tmpfs
size=*nbytes*
Override default maximum size of the filesystem. The size is given in bytes, and rounded up to entire pages. The default is half of the memory. The size parameter also accepts a suffix % to limit this tmpfs instance to that percentage of your physical RAM: the default, when neither size nor nr_blocks is specified, is size=50%

nr_blocks=
The same as size, but in blocks of PAGE_CACHE_SIZE

nr_inodes=
The maximum number of inodes for this instance. The default is half of the number of your physical RAM pages, or (on a machine with highmem) the number of lowmem RAM pages, whichever is the lower.

The tmpfs mount options for sizing (**size**, **nr_blocks**, and **nr_inodes**) accept a suffix **k**, **m** or **g** for Ki, Mi, Gi (binary kilo (kibi), binary mega (mebi) and binary giga (gibi)) and can be changed on remount.

mode=
Set initial permissions of the root directory.

uid= The user id.

gid= The group id.

mpol=[default|prefer:Node|bind:NodeList|interleave|interleave:NodeList]
Set the NUMA memory allocation policy for all files in that instance (if the kernel CONFIG_NUMA is enabled) – which can be adjusted on the fly via 'mount -o remount ...'

default
prefers to allocate memory from the local node

prefer:Node
prefers to allocate memory from the given Node

bind:NodeList
allocates memory only from nodes in NodeList

interleave
prefers to allocate from each node in turn

interleave:NodeList
allocates from each node of NodeList in turn.

The NodeList format is a comma-separated list of decimal numbers and ranges, a range being two hyphen-minus-separated decimal numbers, the smallest and largest node numbers in the range. For example, mpol=bind:0–3,5,7,9–15

Note that trying to mount a tmpfs with an mpol option will fail if the running kernel does not support NUMA; and will fail if its nodelist specifies a node which is not online. If your system relies on that tmpfs being mounted, but from time to time runs a kernel built without NUMA capability (perhaps a safe recovery kernel), or with fewer nodes online, then it is advisable to omit the mpol option from automatic mount options. It can be added later, when the tmpfs is already mounted on MountPoint, by 'mount -o remount,mpol=Policy:NodeList MountPoint'.

Mount options for ubifs

UBIFS is a flash file system which works on top of UBI volumes. Note that **atime** is not supported and is always turned off.

The device name may be specified as

ubiX_Y UBI device number **X**, volume number **Y**

ubiY UBI device number **0**, volume number **Y**

ubiX:NAME
UBI device number **X**, volume with name **NAME**

ubi:NAME
UBI device number **0**, volume with name **NAME**

Alternative **!** separator may be used instead of **:**.

The following mount options are available:

bulk_read
Enable bulk-read. VFS read-ahead is disabled because it slows down the file system. Bulk-Read is an internal optimization. Some flashes may read faster if the data are read at one go, rather than at several read requests. For example, OneNAND can do read-while-load if it reads more than one NAND page.

no_bulk_read
Do not bulk-read. This is the default.

chk_data_crc
Check data CRC-32 checksums. This is the default.

no_chk_data_crc.
Do not check data CRC-32 checksums. With this option, the filesystem does not check CRC-32 checksum for data, but it does check it for the internal indexing information.

This option only affects reading, not writing. CRC-32 is always calculated when writing the data.

compr={none|lzo|zlib}
Select the default compressor which is used when new files are written. It is still possible to read compressed files if mounted with the **none** option.

Mount options for udf

udf is the Universal Disk Format filesystem defined by the Optical Storage Technology Association, and is often used for DVD-ROM. See also*iso9660*.

gid= Set the default group.

umask=
Set the default umask. The value is given in octal.

uid= Set the default user.

unhide
Show otherwise hidden files.

undelete
Show deleted files in lists.

nostrict
Unset strict conformance.

iocharset
Set the NLS character set.

bs= Set the block size. (May not work unless 2048.)

novrs Skip volume sequence recognition.

session=
Set the CDROM session counting from 0. Default: last session.

anchor=
Override standard anchor location. Default: 256.

volume=
Override the VolumeDesc location. (unused)

partition=
Override the PartitionDesc location. (unused)

lastblock=
Set the last block of the filesystem.

fileset=
Override the fileset block location. (unused)

rootdir=
Override the root directory location. (unused)

Mount options for ufs

ufstype=*value*
UFS is a filesystem widely used in different operating systems. The problem are differences among implementations. Features of some implementations are undocumented, so its hard to recognize the type of ufs automatically. That's why the user must specify the type of ufs by mount option. Possible values are:

old Old format of ufs, this is the default, read only. (Don't forget to give the -r option.)

44bsd For filesystems created by a BSD-like system (NetBSD, FreeBSD, OpenBSD).

ufs2 Used in FreeBSD 5.x supported as read-write.

5xbsd Synonym for ufs2.

sun For filesystems created by SunOS or Solaris on Sparc.

sunx86
For filesystems created by Solaris on x86.

hp For filesystems created by HP-UX, read-only.

nextstep
For filesystems created by NeXTStep (on NeXT station) (currently read only).

nextstep-cd
For NextStep CDROMs (block_size == 2048), read-only.

openstep
For filesystems created by OpenStep (currently read only). The same filesystem type is also used by Mac OS X.

onerror=*value*
Set behavior on error:

panic If an error is encountered, cause a kernel panic.

[**lock|umount|repair**]
These mount options don't do anything at present; when an error is encountered only a console message is printed.

Mount options for umsdos

See mount options for msdos. The **dotsOK** option is explicitly killed by *umsdos*.

Mount options for vfat

First of all, the mount options for *fat* are recognized. The **dotsOK** option is explicitly killed by *vfat*. Furthermore, there are

uni_xlate
Translate unhandled Unicode characters to special escaped sequences. This lets you backup and restore filenames that are created with any Unicode characters. Without this option, a '?' is used when no translation is possible. The escape character is ':' because it is otherwise invalid on the vfat filesystem. The escape sequence that gets used, where u is the Unicode character, is: ':', (u & 0x3f), ((u>>6) & 0x3f), (u>>12).

posix Allow two files with names that only differ in case. This option is obsolete.

nonumtail
First try to make a short name without sequence number, before trying *name* ~*num.ext*.

utf8 UTF8 is the filesystem safe 8-bit encoding of Unicode that is used by the console. It can be enabled for the filesystem with this option or disabled with utf8=0, utf8=no or utf8=false. If 'uni_xlate' gets set, UTF8 gets disabled.

shortname=*mode*
Defines the behavior for creation and display of filenames which fit into 8.3 characters. If a long name for a file exists, it will always be the preferred one for display. There are four *mode*s:

lower Force the short name to lower case upon display; store a long name when the short name is not all upper case.

win95 Force the short name to upper case upon display; store a long name when the short name is not all upper case.

winnt Display the short name as is; store a long name when the short name is not all lower case or all upper case.

mixed Display the short name as is; store a long name when the short name is not all upper case. This mode is the default since Linux 2.6.32.

Mount options for usbfs

devuid=*uid* and **devgid=***gid* and **devmode=***mode*
Set the owner and group and mode of the device files in the usbfs filesystem (default: uid=gid=0, mode=0644). The mode is given in octal.

busuid=*uid* and **busgid=***gid* and **busmode=***mode*
Set the owner and group and mode of the bus directories in the usbfs filesystem (default: uid=gid=0, mode=0555). The mode is given in octal.

listuid=*uid* and **listgid=***gid* and **listmode=***mode*
Set the owner and group and mode of the file *devices* (default: uid=gid=0, mode=0444). The mode is given in octal.

Mount options for xenix

None.

Mount options for xfs

allocsize=*size*
Sets the buffered I/O end-of-file preallocation size when doing delayed allocation writeout. Valid values for this option are page size (typically 4KiB) through to 1GiB, inclusive, in power-of-2 increments.

The default behavior is for dynamic end-of-file preallocation size, which uses a set of heuristics to optimise the preallocation size based on the current allocation patterns within the file and the access patterns to the file. Specifying a fixed allocsize value turns off the dynamic behavior.

attr2|noattr2
The options enable/disable an opportunistic improvement to be made in the way inline extended attributes are stored on-disk. When the new form is used for the first time when attr2 is selected (either when setting or removing extended attributes) the on-disk superblock feature bit field will be updated to reflect this format being in use.

The default behavior is determined by the on-disk feature bit indicating that attr2 behavior is active. If either mount option it set, then that becomes the new default used by the filesystem.

CRC enabled filesystems always use the attr2 format, and so will reject the noattr2 mount option if it is set.

barrier|nobarrier
Enables/disables the use of block layer write barriers for writes into the journal and for data integrity operations. This allows for drive level write caching to be enabled, for devices that support write barriers.

discard|nodiscard
Enable/disable the issuing of commands to let the block device reclaim space freed by the filesystem. This is useful for SSD devices, thinly provisioned LUNs and virtual machine images, but may have a performance impact.

Note: It is currently recommended that you use the fstrim application to discard unused blocks rather than the discard mount option because the performance impact of this option is quite severe.

grpid|bsdgroups|nogrpid|sysvgroups
These options define what group ID a newly created file gets. When grpid is set, it takes the group ID of the directory in which it is created; otherwise it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

filestreams
Make the data allocator use the filestreams allocation mode across the entire filesystem rather than just on directories configured to use it.

ikeep|noikeep
When ikeep is specified, XFS does not delete empty inode clusters and keeps them around on disk. When noikeep is specified, empty inode clusters are returned to the free space pool.

inode32|inode64
When inode32 is specified, it indicates that XFS limits inode creation to locations which will not result in inode numbers with more than 32 bits of significance.

When inode64 is specified, it indicates that XFS is allowed to create inodes at any location in the filesystem, including those which will result in inode numbers occupying more than 32 bits of significance.

inode32 is provided for backwards compatibility with older systems and applications, since 64 bits inode numbers might cause problems for some applications that cannot handle large inode numbers. If applications are in use which do not handle inode numbers bigger than 32 bits, the inode32 option should be specified.

largeio|nolargeio
If nolargeio is specified, the optimal I/O reported in st_blksize by stat(2) will be as small as possible to allow user applications to avoid inefficient read/modify/write I/O. This is typically the page size of the machine, as this is the granularity of the page cache.

If largeio specified, a filesystem that was created with a swidth specified will return the swidth value (in bytes) in st_blksize. If the filesystem does not have a swidth specified but does specify an allocsize then allocsize (in bytes) will be returned instead. Otherwise the behavior is the same as if nolargeio was specified.

logbufs=*value*
Set the number of in-memory log buffers. Valid numbers range from 2–8 inclusive.

The default value is 8 buffers.

If the memory cost of 8 log buffers is too high on small systems, then it may be reduced at some cost to performance on metadata intensive workloads. The logbsize option below controls the size of each buffer and so is also relevant to this case.

logbsize=*value*
Set the size of each in-memory log buffer. The size may be specified in bytes, or in kibibytes (KiB) with a k suffix. Valid sizes for version 1 and version 2 logs are 16384 (value=16k) and 32768 (value=32k). Valid sizes for version 2 logs also include 65536 (value=64k), 131072 (value=128k) and 262144 (value=256k). The logbsize must be an integer multiple of the log stripe unit configured at mkfs time.

The default value for version 1 logs is 32768, while the default value for version 2 logs is MAX(32768, log_sunit).

logdev=*device* and **rtdev=***device*
Use an external log (metadata journal) and/or real-time device. An XFS filesystem has up to three parts: a data section, a log section, and a real-time section. The real-time section is optional, and the log section can be separate from the data section or contained within it.

noalign
Data allocations will not be aligned at stripe unit boundaries. This is only relevant to filesystems created with non-zero data alignment parameters (sunit, swidth) by mkfs.

norecovery
The filesystem will be mounted without running log recovery. If the filesystem was not cleanly unmounted, it is likely to be inconsistent when mounted in norecovery mode. Some files or directories may not be accessible because of this. Filesystems mounted norecovery must be mounted read-only or the mount will fail.

nouuid
Don't check for double mounted file systems using the file system uuid. This is useful to mount LVM snapshot volumes, and often used in combination with norecovery for mounting read-only snapshots.

noquota
Forcibly turns off all quota accounting and enforcement within the filesystem.

uquota/usrquota/uqnoenforce/quota
User disk quota accounting enabled, and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

gquota/grpquota/gqnoenforce
Group disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

pquota/prjquota/pqnoenforce
Project disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

sunit=*value* and **swidth=***value*
Used to specify the stripe unit and width for a RAID device or a stripe volume. value must be specified in 512-byte block units. These options are only relevant to filesystems that were created with non-zero data alignment parameters.

The sunit and swidth parameters specified must be compatible with the existing filesystem alignment characteristics. In general, that means the only valid changes to sunit are increasing it by a power-of-2 multiple. Valid swidth values are any integer multiple of a valid sunit value.

Typically the only time these mount options are necessary if after an underlying RAID device has had it's geometry modified, such as adding a new disk to a RAID5 lun and reshaping it.

swalloc
Data allocations will be rounded up to stripe width boundaries when the current end of file is being extended and the file size is larger than the stripe width size.

wsync When specified, all filesystem namespace operations are executed synchronously. This ensures that when the namespace operation (create, unlink, etc) completes, the change to the namespace is on stable storage. This is useful in HA setups where failover must not result in clients seeing inconsistent namespace presentation during or after a failover event.

THE LOOP DEVICE

One further possible type is a mount via the loop device. For example, the command

mount /tmp/disk.img /mnt -t vfat -o loop=/dev/loop3

will set up the loop device */dev/loop3* to correspond to the file */tmp/disk.img*, and then mount this device on */mnt*.

If no explicit loop device is mentioned (but just an option '**-o loop**' is given), then **mount** will try to find some unused loop device and use that, for example

mount /tmp/disk.img /mnt -o loop

The mount command **automatically** creates a loop device from a regular file if a filesystem type is not specified or the filesystem is known for libblkid, for example:

mount /tmp/disk.img /mnt

mount -t ext3 /tmp/disk.img /mnt

This type of mount knows about three options, namely **loop**, **offset** and **sizelimit**, that are really options to losetup(8). (These options can be used in addition to those specific to the filesystem type.)

Since Linux 2.6.25 auto-destruction of loop devices is supported, meaning that any loop device allocated by **mount** will be freed by **umount** independently of */etc/mtab*.

You can also free a loop device by hand, using **losetup -d** or **umount -d**.

RETURN CODES

mount has the following return codes (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).

EXTERNAL HELPERS

The syntax of external mount helpers is:

/sbin/mount.*suffix spec dir* [**-sfnv**] [**-o** *options*] [**-t** *type.subtype*]

where the *suffix* is the filesystem type and the **-sfnvo** options have the same meaning as the normal mount options. The **-t** option is used for filesystems with subtypes support (for example **/sbin/mount.fuse -t fuse.sshfs**).

The command **mount** does not pass the mount options **unbindable**, **runbindable**, **private**, **rprivate**, **slave**, **rslave**, **shared**, **rshared**, **auto**, **noauto**, **comment**, **x-***, **loop**, **offset** and **sizelimit** to the mount.<suffix> helpers. All other options are used in a comma-separated list as argument to the **-o** option.

FILES

/etc/fstab	filesystem table
/etc/mtab	table of mounted filesystems
/etc/mtab~	lock file
/etc/mtab.tmp	temporary file

/etc/filesystems a list of filesystem types to try

ENVIRONMENT

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

SEE ALSO

mount(2), umount(2), fstab(5), umount(8), swapon(8), findmnt(8), nfs(5), **xfs(5),** e2label(8), **xfs_admin(8),** mountd(8), nfsd(8), mke2fs(8), tune2fs(8), losetup(8)

BUGS

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, 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 files */etc/mtab* and */proc/mounts* don't match on systems with 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. remote NFS server. In particular case the mount command may reports unreliable information about a NFS mount point and the /proc/mounts file usually contains more reliable information.) This is another reason to replace mtab file with symlink to the */proc/mounts* file.

Checking files on NFS filesystem referenced by file descriptors (i.e. the **fcntl** and **ioctl** families of functions) may lead to inconsistent result due to the lack of consistency check in kernel even if noac 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** command manually before calling **mount** with the configured loop device.

HISTORY

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

AUTHORS

Karel Zak <kzak@redhat.com>

AVAILABILITY

The mount command is part of the util-linux package and is available from ftp://ftp.kernel.org/pub/linux/utils/util-linux/.