Dateisysteme unter Linux

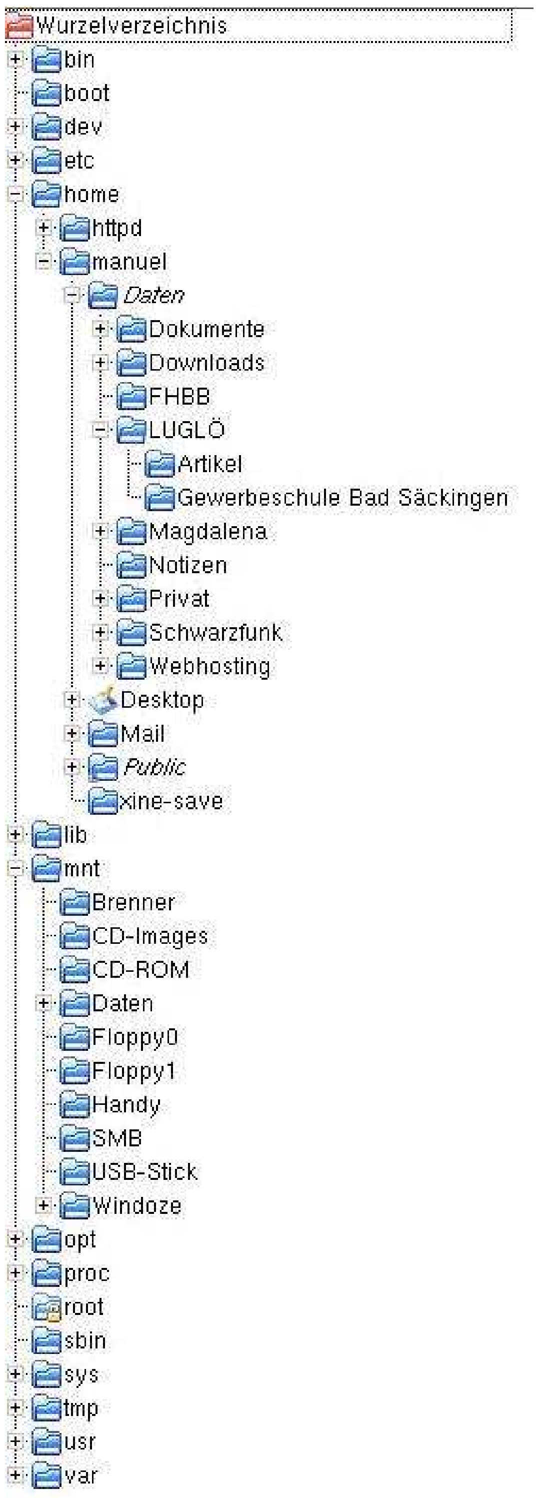
Funktionsweise und Zusammenspiel der Datei /etc/fstab und dem Befehl mount

Inhaltsverzeichnis

Struktur und Funktion des Verzeichnisbaums von Linux:

Linux kennt das von DOS und Windows bekannte System mit verschiedenen Laufwerken die über Laufwerksbuchstaben erreicht werden können nicht. In Linux gibt es nur einen einzigen Verzeichnisbaum der sich von der obersten Ebene aus die Root (Wurzel) oder / genannt wird in verschiedene Verzeichnisse verzweigt.

Bild des Verzeichnisbaumes:



```
Wurzelverzeichnis
+ bin
  boot
+ dev
+ etc
- home
  + httpd
  - manuel
    - Daten
      + Dokumente
      + Downloads
        FHBB
      - LUGLÖ
          Artikel
          Gewerbeschule Bad Säckingen
      + Magdalena
        Notizen
      + Privat
      + Schwarzfunk
      + Webhosting
    + Desktop
    + Mail
    + Public
      xine-save
+ lib
- mnt
    Brenner
    CD-Images
    CD-ROM
  + Daten
    Floppy0
    Floppy1
    Handy
    SMB
    USB-Stick
  + Windoze
+ opt
+ proc
  root
  sbin
+ sys
+ tmp
+ usr
+ var
```

Mounten eines Dateisystems

Wenn man nun auf ein Dateisystem zugreifen möchte muss man als erstes dafür sorgen, dass dieses in dem grossen Verzeichnisbaum enthalten ist.
Diesen Vorgang nennt man „**mounten**“ oder zu deutsch „**einhängen**“.
Dabei wird ein Dateisystem einfach in irgendein vorhandenes Verzeichnis eingehängt, der Inhalt ist danach innerhalb des Verzeichnisses zu finden.

Das Verzeichnis in welches das Dateisystem eingehängt wird heisst „**Mountpoint**“.

Dabei muss das System folgendes wissen:
Welches Dateisystem wird wohin eingehängt, eventuell kommen dazu weitere Optionen.

Beispiel:
```
# mount /dev/hdc /mnt/cdrom
```

Dieser Befehl hängt das Dateisystem des CD-ROM-Laufwerks /dev/hdc in das Verzeichnis /mnt/cdrom ein. Der Inhalt der CD ist danach in /mnt/cdrom zu finden.

Nach der Arbeit mit einem Dateisystem muss dieses auch wieder ausgehängt werden:

Beispiel:
```
# umount /dev/hdc
```
oder
```
# umount /mnt/cdrom
```

Erst nach diesem Schritt darf der Datenträger aus dem Laufwerk entfernt werden!

Um nachzusehen, welche Dateisysteme wo eingehängt sind gibt man folgenden Befehl ein:
```
# mount
```

Beispiel:
```
root@erde# mount
/dev/hda3 on / type reiserfs (rw,noatime)
none on /dev type devfs (rw)
none on /proc type proc (rw)
none on /dev/shm type tmpfs (rw)
```

In der ersten Spalte steht die Partition oder der Datenträger, an zweiter Stelle der Mountpoint, mit type wird das Dateisystem angegeben und in Klammern stehen die Optionen mit denen das Dateisystem gemountet wurde.

Mounten von Dateisystemen die nicht vorher im System eingetragen wurden kann nur root!

Automatisches mounten / Benutzern erlauben zu mounten

Abhilfe schafft die Eintragung von Dateisystem, Mountpoint und Optionen in die Datei /etc/fstab.

Beispiel:
```
/dev/hda1     /boot         ext2      noauto,noatime      1 1
/dev/hda3     /             reiserfs  noatime             0 0
/dev/hda2     none          swap      sw                  0 0
/dev/hdc      /mnt/cdrom    auto      noauto,ro,user      0 0
/dev/fd0      /mnt/floppy   auto      noauto,rw,user      0 0
/dev/sda1     /mnt/memory   auto      noauto,rw,user      0 0
none          /proc         proc      defaults            0 0
none          /dev/shm      tmpfs     defaults            0 0
```

Anhand dieser Eintragungen wird zweierlei gesteuert:
- Welche Dateisysteme beim starten des Computers eingehängt werden und
- welche Dateisysteme dem Benutzer erlaubt sind einzuhängen.

In der ersten Spalte ist wieder der Datenträger oder die Partition angegeben, die zweite Spalte gibt den Mountpoint an, an dritter Stelle wird das Dateisystem angegeben. In der vierten Spalte werden Optionen eingetragen, die beiden letzten Spalten befassen sich mit der Dateisystemprüfung und sollten nicht verändert werden.

Alle Dateisysteme die hier eingetragen sind werden beim Start des Computers eingehängt – es sei denn, ein **`noauto`** verhindert dies. Dann wird das Dateisystem nicht eingehängt, aber es kann später einfach eingehängt werden, wenn ebenfalls **`user`** in den Optionen enthalten ist auch von normalen Benutzern.
Bei Dateisystemen die in der Datei fstab eingetragen sind kann man der Einfachheit halber auch nur Partition/Datenträger oder Mountpoint angeben, den Rest sucht sich mount dann aus.

Beispiel:
```
# mount /dev/fd0
```
macht dasselbe wie
```
# mount /mnt/floppy
```
oder
```
# mount /dev/fd0 /mnt/floppy
```

Wollen wir nun dafür sorgen, dass beim Start des Computers unsere Windows-Festplatte eingehängt wird welche die erste Partition der zweiten Festplatte darstellt, so müssen wir folgende Eintragung hinzufügen:

```
/dev/hdb1     /mnt/windows   vfat gid=100,umask=007      0 0
```

Bedeutung:

`/dev/hdb1`:	hdb=zweite Festplatte, 1=erste Partition
`/mnt/windows:`	Beliebiges existierendes Verzeichnis, hier /mnt/windows.
`vfat`:	Dateisystemtyp. vfat wurde bei DOS und Windows 9x verwendet.
`gid=100,umask=007`:	Optionen:

Linux-Benutzerberechtigungen für fremde Dateisysteme:

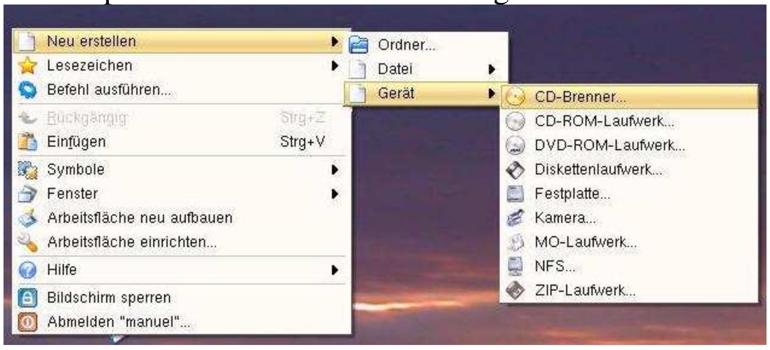
Da vfat keine Linux Benutzerrechte kennt werden beim Einhängen des Dateisystems übergeordnete Rechte für den gesamten Inhalt des Dateisystems vergeben. Diese orientieren sich am Benutzer der das Dateisystem einhängt – beim automatischen Einhängen beim Start des Computers ist das root, der Administrator.
Somit hat nachher der Benutzer keinen Schreibzugriff auf das Laufwerk. Um dies zu ändern setzen wir die Gruppenzugehörigkeit des Dateisystems mit gid=100 auf die Gruppe Nr. 100 – users.
Die Benutzerberechtigungen setzen wir mit umask=007 auf:
- Vollzugriff für den Besitzer (root)
- Vollzugriff für die Besitzergruppe (gid=100; users)
- keine Rechter für alle anderen Benutzer

Somit haben wir erreicht was wir wollen.

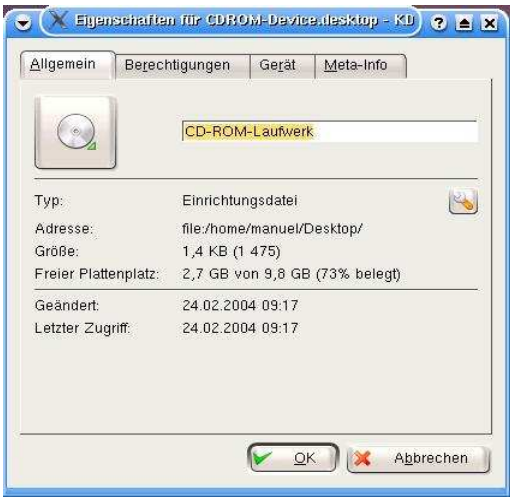
Bei Dateisystemen welche der Benutzer selbst mounten und unmounten darf (muss, z.B. bei CDs oder Disketten) müssen wir diesen Aufwand nicht betreiben. Wer die Manpage von mount liest wird feststellen, dass man solche Optionen nämlich auch für iso9660 findet, dem Dateisystem für CDs.
Da die Benutzerzugehörigkeit der Daten immer auf den Benutzer gesetzt werden, der das Dateisystem einhängt werden wir damit nie Probleme haben, wenn wir als Benutzer selber einhängen.

<u>Erstellen von Symbolen in KDE zum Ein- und Aushängen von Dateisystemen:</u>

Rechtsklick auf dem Desktop -> Neu erstellen -> Gerät -> gewünschtes Auswählen



Dem Symbol einen Namen geben -> Gerät



Eigenschaften für CDROM-Device.desktop - KD

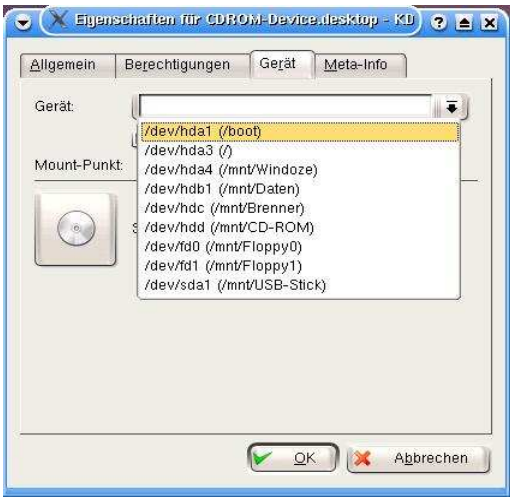
Allgemein | Berechtigungen | Gerät | Meta-Info

CD-ROM-Laufwerk

Typ:	Einrichtungsdatei
Adresse:	file:/home/manuel/Desktop/
Größe:	1,4 KB (1 475)
Freier Plattenplatz:	2,7 GB von 9,8 GB (73% belegt)
Geändert:	24.02.2004 09:17
Letzter Zugriff:	24.02.2004 09:17

OK | Abbrechen

Gerät auswählen -> OK



Eigenschaften für CDROM-Device.desktop - KD

Allgemein | Berechtigungen | Gerät | Meta-Info

Gerät:

Mount-Punkt:

```
/dev/hda1 (/boot)
/dev/hda3 (/)
/dev/hda4 (/mnt/Windoze)
/dev/hdb1 (/mnt/Daten)
/dev/hdc (/mnt/Brenner)
/dev/hdd (/mnt/CD-ROM)
/dev/fd0 (/mnt/Floppy0)
/dev/fd1 (/mnt/Floppy1)
/dev/sda1 (/mnt/USB-Stick)
```

OK | Abbrechen

Ab sofort wird das Dateisystem beim Klick auf das entsprechende Symbol eingehängt und angezeigt.
Zum Aushängen einfach Rechtsklick – Laufwerk aushängen auswählen.

<u>Auszug aus der Manpage von fstab:</u>
Befehl: `# man fstab`

DESCRIPTION

The file fstab contains descriptive information about the various file systems. fstab is only read by programs, and not written; it is the duty of the system administrator to properly create and maintain this file. Each filesystem is described on a separate line; fields on each line are separated by tabs or spaces. Lines starting with '#' are comments. The order of records in fstab is important because fsck(8), mount(8), and umount(8) sequentially iterate through fstab doing their thing.

The first field, (fs_spec), describes the block special device or remote filesystem to be mounted.

For ordinary mounts it will hold (a link to) a block special device node (as created by mknod(8)) for the device to be mounted, like `/dev/cdrom' or `/dev/sdb7'. For NFS mounts one will have <host>:<dir>, e.g., `knuth.aeb.nl:/'. For procfs, use `proc'.

Instead of giving the device explicitly, one may indicate the (ext2 or xfs) filesystem that is to be mounted by its UUID or volume label (cf. e2label(8) or xfs_admin(8)), writing LABEL=<label> or UUID=<uuid>, e.g., `LABEL=Boot' or `UUID=3e6be9de-8139-11d1-9106-a43f08d823a6'. This will make the system more robust: adding or removing a SCSI disk changes the disk device name but not the filesystem volume label.

The second field, (fs_file), describes the mount point for the filesystem. For swap partitions, this field should be specified as `none'. If the name of the mount point contains spaces these can be escaped as `\040'.

The third field, (fs_vfstype), describes the type of the filesystem. Linux supports lots of filesystem types, such as adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, and possibly others. For more details, see mount(8). For the filesystems currently supported by the running kernel, see /proc/filesystems. An entry swap denotes a file or partition to be used for swapping, cf. swapon(8). An entry ignore causes the line to be ignored. This is useful to show disk partitions which are currently unused.

The fourth field, (fs_mntops), describes the mount options associated with the filesystem.

It is formatted as a comma separated list of options. It contains at least the type of mount plus any additional options appropriate to the filesystem type. For documentation on the available options for non-nfs file systems, see mount(8). For documentation on all nfs-specific options have a look at nfs(5). Common for all types of file system are the options ``noauto'' (do not mount when "mount -a" is given, e.g., at boot time), ``user'' (allow a user to mount), and ``owner'' (allow device owner to mount), and ``_netdev'' (device requires network to be available). The ``owner'' and ``_netdev'' options are Linux-specific. For more details, see mount(8).

The fifth field, (fs_freq), is used for these filesystems by the dump(8) command to determine which filesystems need to be dumped. If the fifth field is not present, a value of zero is returned and dump will assume that the filesystem does not need to be dumped.

The sixth field, (fs_passno), is used by the fsck(8) program to determine the order in which filesystem checks are done at reboot time. The root filesystem should be specified with a fs_passno of 1, and other filesystems should have a fs_passno of 2. Filesystems within a drive will be checked sequentially, but filesystems on different drives will be checked at the same time to utilize parallelism available in the hardware. If the sixth field is not present or zero, a value of zero is returned and fsck will assume that the filesystem does not need to be checked.

Auszug aus der Manpage von mount
Befehl: `# man mount`

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 file system 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 type device dir
```

This tells the kernel to attach the file system 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 file system remains mounted, the pathname dir refers to the root of the file system on device.

The file /etc/fstab (see fstab(5)), may contain lines describing what devices are usually mounted where, using which options. This file is used in three ways:

(i) The command

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

(usually given in a bootscript) causes all file systems 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.

(ii) When mounting a file system mentioned in fstab, it suffices to give only the device, or only the mount point.

(iii) Normally, only the superuser can mount file systems. However, when fstab contains the user option on a line, then anybody can mount the corresponding system.

Thus, given a line

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

any user can mount the iso9660 file system found on his 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, 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 programs mount and umount maintain a list of currently mounted file systems in the file /etc/mtab. If no arguments are given to mount, this list is printed.

When the proc filesystem is mounted (say at /proc), the files /etc/mtab and /proc/mounts have very similar contents. The former has somewhat more information, such as the mount options used, but is not necessarily up-to-date (cf. the -n option below). It is possible to replace /etc/mtab by a symbolic link to /proc/mounts, but some information is lost that way, and in particular working with the loop device will be less convenient, and using the "user" option will fail.

OPTIONS

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

Options available for the mount command:

-h Print a help message.

-v Verbose mode.

-a Mount all filesystems (of the given types) mentioned in fstab.

-r Mount the file system read-only. A synonym is -o ro.

-w Mount the file system read/write. This is the default. A synonym is -o rw.

-t vfstype

The argument following the -t is used to indicate the file system type. The file system types which are currently supported are: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, ramfs, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, xiafs. Note that coherent, sysv and xenix are equivalent and that xenix and coherent will be removed at some point in the future -- use sysv instead. Since kernel version 2.1.21 the types ext and xiafs do not exist anymore.

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 if you use a kernel module autoloader. Warning: the probing uses a heuristic (the presence of appropriate `magic'), and could recognize the wrong filesystem type, possibly with catastrophic consequences. If your data is valuable, don't ask mount to guess.

-o Options are specified with a -o flag followed by a comma separated string of options. Some of these options are only useful when they appear in the /etc/fstab file. The following options apply to any file system that is being mounted (but not every file system actually honors them - e.g., the sync option today has effect only for ext2, ext3 and ufs):

async All I/O to the file system should be done asynchronously.

atime Update inode access time for each access. This is the default.

auto Can be mounted with the -a option.

defaults

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

dev Interpret character or block special devices on the file system.

exec Permit execution of binaries.

_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).

noatime
Do not update inode access times on this file system (e.g, for faster access on the news spool to speed up news servers).

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

nodev Do not interpret character or block special devices on the file system.

noexec Do not allow execution of any binaries on the mounted file system. This option might be useful for a server that has file systems containing binaries for architectures other than its own.

nosuid Do not allow set-user-identifier or set-group-identifier bits to take effect. (This seems safe, but is in fact rather unsafe if you have suidperl(1) installed.)

nouser Forbid an ordinary (i.e., non-root) user to mount the file system. This is the default.

remount
Attempt to remount an already-mounted file system. This is commonly used to change the mount flags for a file system, especially to make a readonly file system writeable. It does not change device or mount point.

ro Mount the file system read-only.

rw Mount the file system read-write.

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

sync All I/O to the file system should be done synchronously.

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

user Allow an ordinary user to mount the file system. The name of the mounting user is written to mtab so that he can unmount the file system again. This option implies the options noexec, nosuid, and nodev (unless overridden by subsequent options, as in the option line user,exec,dev,suid).

users Allow every user to mount and unmount the file system. This option implies the options noexec, nosuid, and nodev (unless overridden by subsequent options, as in the option line users,exec,dev,suid).

--bind Remount a subtree somewhere else (so that its contents are available in both places).

--move Move a subtree to some other place. See above.

FILESYSTEM SPECIFIC MOUNT OPTIONS

The following options apply only to certain file systems. We sort them by file system. They all follow the -o flag.

Mount options for ext2

The `ext2' file system is the standard Linux file system. Due to a kernel bug, it may be mounted with random mount options (fixed in Linux 2.0.4).

bsddf / minixdf

Set the behaviour for the statfs system call. The minixdf behaviour is to return in the f_blocks field the total number of blocks of the file system, while the bsddf behaviour (which is the default) is to subtract the overhead blocks used by the ext2 file system 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 / check=normal / check=strict

Set checking level. When at least one of these options is set (and check=normal is set by default) the inodes and blocks bitmaps are checked upon mount (which can take half a minute or so on a big disk, and is rather useless). With strict checking, block deallocation checks that the block to free is in the data zone.

check=none / nocheck

No checking is done. This is fast. Recent kernels do not have a check option anymore - checking with e2fsck(8) is more meaningful.

debug Print debugging info upon each (re)mount.

errors=continue / errors=remount-ro / errors=panic

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

grpid or bsdgroups / nogrpid or 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.

resgid=n and resuid=n
The ext2 file system 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 hundreds or even 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 1k units. Thus, if you want to use logical block 32768 on a filesystem with 4k blocks, use "sb=131072".

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

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

Mount options for ext3

The `ext3' file system is version of the ext2 file system which has been enhanced with journalling. It supports the same options as ext2 as well as the following additions:

journal=update
Update the ext3 file system'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 file system's journal file; ext3 will create a new journal, overwriting the old contents of the file whose inode number is inum.

noload Do not load the ext3 file system's journal on mounting.

data=journal / data=ordered / data=writeback

Specifies the journalling mode for file data. Metadata is always journaled.

journal

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

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 file system after its metadata has been committed to the journal. This is rumoured to be the highest-throughput option. It guarantees internal file system integrity, however it can allow old data to appear in files after a crash and journal recovery.

Mount options for fat, msdos, vfat

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

blocksize=512 / blocksize=1024 / blocksize=2048

Set blocksize (default 512).

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. Present since 2.5.43.

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. Present since 2.5.43.

check=value

Three different levels of pickyness 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 may not contain long parts and special characters that are sometimes used on Linux, but are not accepted by MS-DOS are rejected. (+, =, spaces, etc.)

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

conv=b[inary] / conv=t[ext] / conv=a[uto]
The fat file system can perform CRLF<-->NL (MS-DOS text format to UNIX text format) conversion in the kernel. The following conversion modes are available:

binary no translation is performed. This is the default.

text CRLF<-->NL translation is performed on all files.

auto 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 file systems mounted in binary mode, a conversion tool (fromdos/todos) is available.

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.

cvf_option=option
Option passed to the CVF module.

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

fat=12 / fat=16 / fat=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.

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

sys_immutable, showexec, dots, nodots, dotsOK=[yes|no]
Various misguided attempts to force Unix or DOS conventions onto a FAT file system.

Mount options for ntfs

iocharset=name
Character set to use when returning file names. Unlike VFAT, NTFS suppresses names that contain unconvertible characters.

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 file system distinguishes between upper and lower case. The 8.3 alias names are presented as hard links instead of being suppressed.

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 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 file system (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] / check=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 file system the indicated user or group id, possibly overriding the information found in the Rock Ridge extensions. (Default: uid=0,gid=0.)

map=n[ormal] / map=o[ff] / map=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 like map=normal but also apply Acorn extensions if present.

mode=value
For non-Rock Ridge volumes, give all files the indicated mode. (Default: read 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: block=1024.)

conv=a[uto] / conv=b[inary] / conv=m[text] / conv=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 16MB. The `cruft' option is set automatically if the entire CDROM has a weird size (negative, or more than 800MB). It is also set when volume sequence numbers other than 0 or 1 are seen.
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.

<u>Glossar:</u>

Bootsektor	Kleiner Bereich auf dem >*Datenträger* (ein >*Sektor* gross) der dafür gemacht ist um ein kleines Ladeprogramm zu enthalten welches beim Start des Rechners ein Betriebssystem starten kann. Ein Bootsektor ist nicht im >*Dateisystem* enthalten, das heisst man kann als Benutzer nicht auf ihn zugreifen, nur die entsprechenden Ladeprogramme greifen darauf zu um sich dort zu speichern. Ein vorhandener Bootsektor muss nicht zwangsläufig beschrieben sein, dann ist allerdings der >*Datenträger* oder die >*Partition* nicht bootbar, also startfähig. Die bekanntesten Ladeprogramme für Linux sind LILO oder Grub.
Dateisystem	Um Daten, welche in eine >*Partition* abgelegt werden und dort auch wiedergefunden werden sollen, in eine entsprechende Struktur zu bringen benötigt man ein Dateisystem, ein Verzeichnis in welchem das Betriebssystem speichert wo welche Datei und welcher Ordner zu finden ist. Im Dateisystem ist das enthalten was der Benutzer sieht wenn er mit Dateien und Ordnern arbeitet. In manchen Dateisystemen werden ausserdem Benutzerberechtigungen oder verschiedene Attribute gespeichert wie beispielsweise „versteckt“ oder „schreibgeschützt“. Jedes Dateisystem bietet seine eigenen Vor- und Nachteile, manche sind besonders stabil was Datenverlust angeht, andere besonders schnell oder besonders sicher. Beispiele: fat, vfat, msdos — Windowsdateisystem ohne Benutzerberechtigungen (DOS, Win9x) ntfs — Windows Dateisystem mit Benutzerberechtigungen (NT, 2000, XP) ext2 — altes Linux Dateisystem ext3 — neues Linux Dateisystem, kann sich regenerieren reiserfs — sehr schnelles und stabiles Linux Dateisystem iso9660 — Dateisystem für CD-ROMs
Datenträger	Der Datenträger ist das Medium (Hardware) welches die gespeicherten Daten enthält. Ein Datenträger kann entweder ein >*Laufwerk* sein welches fest im Computer eingebaut wird (Festplatte) oder ein magnetisches (Diskette, Band), optisches (CD, DVD) oder elektronisches Medium (SDCard, USB-Stick) sein. Es gibt zwei Gruppen von Datenträgern: Festplatten welche fest in ihrem Laufwerk eingebaut sind und Wechseldatenträger die dafür gemacht sind um zwischen Computern auszutauschen. Diese müssen meist zur Benutzung in entsprechende >*Laufwerke* eingeführt werden (Disketten, CDs, DVDs). Manche moderne Wechseldatenträger können direkt über übliche PC-Schnittstellen angeschlossen werden (USB-Stick).

Bootsektor	Kleiner Bereich auf dem >*Datenträger* (ein >*Sektor* gross) der dafür gemacht ist um ein kleines Ladeprogramm zu enthalten welches beim Start des Rechners ein Betriebssystem starten kann. Ein Bootsektor ist nicht im >*Dateisystem* enthalten, das heisst man kann als Benutzer nicht auf ihn zugreifen, nur die entsprechenden Ladeprogramme greifen darauf zu um sich dort zu speichern. Ein vorhandener Bootsektor muss nicht zwangsläufig beschrieben sein, dann ist allerdings der >*Datenträger* oder die >*Partition* nicht bootbar, also startfähig. Die bekanntesten Ladeprogramme für Linux sind LILO oder Grub.
Laufwerk	Das elektrische Gerät welches im Computer eingebaut ist oder extern an diesen angeschlossen werden kann in dem >*Datenträger* eines bestimmten Typs gelesen (und meist auch beschrieben) werden können. Unterschieden wird zwischen Laufwerke für Wechsel>*datenträger* (CD-ROM, DVD, Disketten) und anderen >*Datenträger* die fest im Laufwerk enthalten sind: Festplatten.
Mountpount	Ein Verzeichnis in dem man ein >*Dateisystem* einhängt. Um in UNIX auf ein >*Dateisystem* zugreifen zu können muss es in den zentralen Verzeichnisbaum eingehängt – gemountet – werden. Somit befindet sich dann der Inhalt des >*Dateisystems* unterhalb des Mountpoints, des Verzeichnis in das das >*Dateisystem* eingehängt wurde.
Partition	Eine Partition ist eine Abschnitt eines >*Datenträgers* der ein >*Dateisystem* enthält. Ein >*Datenträger* kann vier Partitionen enthalten (primäre Partitionen genannt). Wenn das nicht ausreicht, kann eine erweiterte Partition wiederum vier logische Partitionen enthalten. In diesem Fall ist die erweiterte Partition nicht für ein >*Dateisystem* nutzbar, sie ist lediglich der Container für bis zu vier weitere Partitionen. Jede primäre Partition enthält einen eigenen >*Bootsektor*.