Learn Linux, 101: Control mounting and unmounting of filesystems

Getting to your data

Ian Shields
Senior Programmer
IBM

20 October 2010

Learn to mount your Linux® filesystems; configure and use removable USB, IEE 1394, or other devices; and properly access floppy disks, CDs, and DVDs. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to learn about accessing Linux filesystems.

View more content in this series

Overview

In this article, learn to:

- Mount and unmount filesystems manually
- Configure filesystem mounting on bootup
- Configure user-mountable, removable filesystems such as tape drives, external USB drives, floppies, and CDs or DVDs

This article helps you prepare for Objective 104.3 in Topic 104 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 3.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. Unless otherwise noted, the examples in this article use Fedora 13, with a 2.6.34 kernel. Your results on other systems may differ. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here.

You should also be familiar with the material in our article, "Learn Linux 101: Create partitions and filesystems."

Mounting filesystems

Connect with Ian

Ian is one of our most popular and prolific authors. Browse all of Ian's articles on developerWorks. Check out Ian's profile and connect with him, other authors, and fellow readers in My developerWorks.

The Linux filesystem is one big tree rooted at /, and yet we have filesystems on different devices and partitions. How do we resolve this apparent incongruity? The root (/) filesystem is mounted as part of the initialization process. Each of the other filesystems that you create is not usable by your Linux system until it is *mounted* at a *mount point*.

In the current set of mounted filesystems, a mount point is simply a directory where the filesystem on a device is grafted into the tree. Mounting is the process of making the filesystem on the device accessible. For example, you might mount filesystems on hard drive partitions as /boot, /tmp, or /home, and you might mount the filesystem on a floppy drive as /mnt/floppy and the filesystem on a CD-ROM as /media/cdrom1. As you can see, mount points may be in the root directory, or in a subdirectory farther down the tree.

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our developerWorks roadmap for LPIC-1 for a description of and link to each article in this series. The roadmap is in progress and reflects the *current objectives (April 2009)* for the LPIC-1 exams. As we complete articles, we add them to the roadmap. In the meantime, in our previous LPI certification exam prep tutorials, find earlier versions of similar material that supports LPIC-1 *objectives prior to April 2009*.

Besides filesystems on partitions, floppy disks, and CDs, there are other types of filesystems. The tmpfs filesystem is a virtual memory filesystem. You can also mount filesystems from one system on another system using a networked filesystem such as NFS or AFS. You can even create a file in an existing filesystem and format that as a, possibly different, kind of filesystem and mount that too. This is often done with images of optical media, where you perhaps download an ISO CD or DVD image, then mount that file rather than burning it to real media. Swap space in a file rather than a dedicated swap partition is another example.

While the mount process actually mounts the *filesystem* on some device (or other resource), it is common to simply say that you "mount the device," which is understood to mean "mount the filesystem on the device."

Mounting and unmounting filesystems usually requires root authority. If you are logged in as an ordinary user, you will either use `su -` to switch to root or `sudo`. In our examples, when the command prompt ends with #, as in Listing 1 below, you will need root authority.

The basic form of the `mount` command takes two parameters: the device (or other resource) containing the filesystem to be mounted, and the mount point. For example, we mount our FAT32 partition /dev/sda9 at the mount point /dos as shown in Listing 1.

Listing 1. Mounting /dos

```
[root@echidna ~]# mount /dev/sda9 /dos
```

The mount point must exist before you mount anything over it. If it does not, you will get an error and need to create the mount point or use a different mount point as illustrated in Listing 2.

Listing 2. Mount error

```
[root@echidna ~]# mount /dev/sda9 /dos
mount: mount point /dos does not exist
[root@echidna ~]# mkdir /dos
[root@echidna ~]# mount /dev/sda9 /dos
```

When you mount a filesystem over an existing directory, the files on the filesystem you are mounting become the files and subdirectories of the mount point. If the mount point directory already contained files or subdirectories, they are not lost, but are no longer visible until the mounted filesystem is unmounted, at which point they become visible again. It is a good idea to avoid this problem by using only empty directories as mount points.

After mounting a filesystem, any files or directories created or copied to the mount point or any directory below it will be created on the mounted filesystem. So a file such as /dos/sampdir/file.txt will be created on the FAT32 filesystem that we mounted at /dos in our example.

Usually, the `mount` command will automatically detect the type of filesystem being mounted. Occasionally you may need to specify the filesystem type explicitly using the `-t` option as shown in Listing 3.

Listing 3. Mounting with explicit filesystem type

```
[root@echidna ~]# mount -t vfat /dev/sda9 /dos
```

To see what filesystems are mounted, use the `mount` command with no parameters. Listing 4 shows our example system. Note that you do not need root authority to simply list mounted filesystems.

Listing 4. Displaying mounted filesystems

```
[ian@echidna ~]$ mount
/dev/sda6 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda2 on /grubfile type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
gvfs-fuse-daemon on /home/ian/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev,user=ian)
dw.raleigh.ibm.com:/vol/vol1/dwcontent on /mnt/dwcontent type nfs (rw,addr=9.42.155.6)
/dev/sdb9 on /mnt/sdb9 type ext3 (rw)
/dev/sda9 on /dos type vfat (rw)
/dev/sr0 on /media/KNOPPIX type iso9660 (ro,nosuid,nodev,uhelper=udisks,uid=1000,gid=1000
,iocharset=utf8,mode=0400,dmode=0500)
```

You can also view similar information by displaying /proc/mounts or /etc/mtab, both of which contain information about mounted filesystems.

Mount options

The `mount` command has several options that override the default behavior. For example, you can mount a filesystem read-only by specifying `-o ro`. If the filesystem is already mounted, add `remount` as shown in Listing 5.

Listing 5. Remounting read-only

```
[root@echidna ~]# mount -o remount,ro /dos
```

Notes:

- Separate multiple options, such as `remount` and `ro`, with commas.
- When remounting an already mounted filesystem, it suffices to specify either the mount point or the device name. It is not necessary to specify both.
- You cannot mount a read-only filesystem as read-write. Media that cannot be modified, such as CD-ROM discs, will automatically be mounted read-only.
- To remount a writable device read-write, specify `-o remount,rw`

Remount commands will not complete successfully if any process has open files or directories in the filesystem being remounted. See Unmounting filesystems below for additional information.

Labels, UUIDs, and links

In UNIX and early Linux systems, the /dev directory usually contained entries for all the devices that might ever be attached to a system. Any device that was used was always located in the same place in the /dev tree, so using names such as /dev/sda6 was natural. With the advent of hot-plugging of devices such as USB or Firewire (IEEE 1394) attached devices, a given device might appear in one USB port today, and that same device might be plugged into a different USB port tomorrow. In this environment, you might want to always mount your USB stick at /media/myusbstick, regardless of which USB port you plug it in to. In the article for topic 102, "Learn Linux, 101: Boot managers," you learned about using labels and UUIDs (Universally Unique IDs) instead of device names to identify partitions. If the filesystem on the partition supports either, you can use these with the `mount` command too. Use the `blkid` command to find out the UUID and label (if present) associated with a device. Listing 6 shows how to use `blkid` to find the label and UUID for our root partition and then how to create two additional mount points and mount the root partition at these two additional points. This example is for illustration. You would not normally do this in a production environment.

Listing 6. Mount using labels or UUIDs

```
[root@echidna ~]# blkid /dev/sda6
/dev/sda6: LABEL="Fedora-13-x86_64" UUID="082fb0d5-a5db-41d1-ae04-6e9af3ba15f7"
 TYPE="ext4"
[root@echidna ~]# mkdir /mnt/sda6label
[root@echidna ~]# mkdir /mnt/sda6uuid
[root@echidna ~]# mount LABEL="Fedora-13-x86_64" /mnt/sda6label
[root@echidna ~]# mount UUID="082fb0d5-a5db-41d1-ae04-6e9af3ba15f7" /mnt/sda6uui
```

With the advent of udev, you will usually find additional symbolic links in the /dev directory for devices such as hard drives. Listing 7 shows the links to /dev/sda6 on my Fedora 13 system.

Listing 7. Symbolic links to /dev/sda6

```
[ian@echidna ~]$ find /dev -lname "*sda6"
/dev/disk/by-label/Fedora-13-x86_64
/dev/disk/by-uuid/082fb0d5-a5db-41d1-ae04-6e9af3ba15f7
/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part6
/dev/disk/by-id/wwn-0x50014ee001a8d027-part6
/dev/disk/by-id/scsi-SATA_WDC_WD1001FALS-_WD-WMATV3772868-part6
/dev/disk/by-id/ata-WDC_WD1001FALS-00J7B1_WD-WMATV3772868-part6
/dev/block/8:6
```

You can also use a symbolic link as another way of specifying the device name when mounting a device.

Boot time and fstab

In the article for topic 102, "Learn Linux, 101: Boot managers," you learned how to use the `root=` parameter in both GRUB and LILO to tell the boot loader what filesystem should be mounted as root. Once this filesystem is mounted, the initialization process runs `mount` with the `-a` option to automatically mount a set of filesystems. The set is specified in the file /etc/fstab. Listing 8 shows /etc/fstab for a sample Fedora 13 system. In this example, all hard drive partitions are identified by UUID. Listing 9 shows a different example from an Ubuntu 9.10 system.

Listing 8. Example Fedora 13 fstab

```
#
# /etc/fstab
# Created by anaconda on Fri May 28 12:37:05 2010
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=082fb0d5-a5db-41d1-ae04-6e9af3ba15f7 /                  ext4    defaults        1 1
UUID=488edd62-6614-4127-812d-cbf58eca85e9 /grubfile          ext3    defaults        1 2
UUID=2d4f10a6-be57-4e1d-92ef-424355bd4b39 swap               swap    defaults        0 0
UUID=ba38c08d-a9e7-46b2-8890-0acda004c510 swap               swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

Listing 9. Example Ubuntu 9.10 fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/sda7 during installation
UUID=8954fa66-e11f-42dc-91f0-b4aa480fa103 /               ext3    errors=remount-ro 0  1
# /grubfile was on /dev/sda2 during installation
UUID=3a965842-b6dd-4d52-8830-2d0fdb4284a2 /grubfile       ext3    defaults        0  2
/dev/sda5       none            swap    sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto,exec,utf8 0        0
/dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0        0
```

Lines starting with a # character are comments. Remaining lines contain six fields. Because the fields are positional, they must all be specified.

file system
This may be a device name such as /dev/sda1, or a label (LABEL=) or UUID (UUID=). For the root filesystem of our Fedora 13 example, it could be /dev/sda6, LABEL="Fedora-13-x86_64", or UUID="082fb0d5-a5db-41d1-ae04-6e9af3ba15f7". Using a label or UUID makes your system more robust when devices are added or removed.

mount point
This is the mount point we discussed in Mounting filesystems above. For swap space, this should be the value 'none' or 'swap'. On older systems you will usually find the value 'none'.

type
Specifies the type of filesystem. CD/DVD drives will often support either ISO9660 or UDF filesystems, so you may specify multiple possibilities in a comma-separated list as you see in Listing 9. If you want `mount` to automatically determine the type, specify `auto` as is done in the last line of Listing 9 for the floppy drive.

option
Specifies the mount options. Specify `defaults` if you want default mount options. Some options you will want to know about are:

- `rw` and `ro` specify whether the filesystem should be mounted read-write or read-only.
- `noauto` specifies that this filesystem should not be automatically mounted at boot time or whenever `mount -a` is issued. In our example, this is done for the removable drives.
- `user` specifies that a non-root user is permitted to mount and unmount the filesystem. This is especially useful for removable media. In older systems, this option is specified in /etc/fstab rather than on the `mount` command. With newer systems, it may be specified in udev rules that are located in rules files within /lib/udev/rules.d or /etc/udev/rules.d. The options for the DVD drive on my Fedora 13 system come from udev rules, and that is why there is no entry in /etc/fstab for an optical drive.
- `exec` and `noexec` specify whether or not to allow execution of files from the mounted filesystem. User-mounted filesystems default to `noexec` unless `exec` is specified **after**`user`.

- `noatime` will disable recording of access times. Not using access times may improve performance.

dump
Specifies whether the `dump` command should consider this ext2 or ext3 filesystem for backups. A value of 0 tells `dump` to ignore this filesystem.

pass
Non-zero values of pass specify the order of checking filesystems at boot time, as discussed in our article "Learn Linux, 101: Maintain the integrity of filesystems."

When you mount a filesystems that is listed in /etc/fstab, you can give either the device name or the mount point when mounting the filesystem. You do not need to give both.

On some systems, for example SUSE 11.2, you may find that the `fstab` generated at install time uses symbolic links to the device. So, you may see /dev/disk/by-id/ata-WDC_WD1001FALS-00J7B1_WD-WMATV3772868-part6, rather than /dev/sda6 for the file system value.

Consult the man pages for `fstab`, `mount`, and `udev` for additional information, including options not covered here.

Unmounting filesystems

All mounted filesystems are usually unmounted automatically by the system when it is rebooted or shut down. When a filesystem is unmounted, any cached filesystem data in memory is flushed to the disk.

You may also unmount filesystems manually. Indeed, you **should** do this when removing writable media such as diskettes or USB drives or memory keys.

Use the `umount` command to unmount the filesystem, specifying *either* the device name or mount point as an argument. Listing 10 shows how to unmount /dos, then remount it and unmount again using the device name.

Listing 10. Unmounting filesystems

```
[root@echidna ~]# umount /dos
[root@echidna ~]# mount /dev/sda9 /dos
[root@echidna ~]# umount /dev/sda9
```

After a filesystem is unmounted, any files in the directory used for the mount point are visible again.

If you attempt to unmount a filesystem while a process has open files on that filesystem, you will see an error message. Before unmounting a filesystem, you should check that there are no processes running that have open files on the filesystem. Use the `lsof` or `fuser`command to determine what files are open or what process has open files. You may need the `-w` option on `lsof` to avoid warning messages related to the Gnome Virtual File system (gvfs). Check the man pages to learn about additional mount options and `lsof`. If you are checking a whole device, you can specify the device name or the mount point. You may also check whether an individual file is in use or not.

To illustrate these commands, I created a copy of /etc/fstab on /dos and a small script to read lines from stdin and print them to stdout with a 10 second pause between each line. Listing 11 shows the error message from `umount` when files are in use and the use of `lsof` and `fuser` to check for open files on /dos, or the underlying device /dev/sda9.

Listing 11. Checking for open files

```
[root@echidna ~]# umount /dos
umount: /dos: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
[root@echidna ~]# lsof -w /dos
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
slowread. 2560  ian    0r   REG    8,9      899  123 /dos/fstab
sleep     2580  ian    0r   REG    8,9      899  123 /dos/fstab
[root@echidna ~]# lsof -w /dev/sda9
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
slowread. 2560  ian    0r   REG    8,9      899  123 /dos/fstab
sleep     2580  ian    0r   REG    8,9      899  123 /dos/fstab
[root@echidna ~]# fuser -m /dos
/dos:                 2560  2586
[root@echidna ~]# fuser -m /dev/sda9
/dev/sda9:            2560  2588
```

At this point you can either wait until the filesystem is no longer busy, or you can do a *lazy unmount*, by specifying the `-l` option. A lazy unmount detaches the filesystem from the filesystem tree immediately, and cleans the references to the filesystem when it is no longer busy.

Removable filesystems

We mentioned some issues with removable devices such as USB or Firewire (IEEE 1394) attached devices. It is inconvenient to switch to root access every time you need to mount or unmount such a device. The same goes for CD, DVD, and floppy drives, where you need to unmount the device to change media. In the discussion of `fstab` above, we mentioned the `user` option, which allows ordinary users to mount and unmount devices. Listing 9 shows one way to code `fstab` entries for a floppy drive or for a CD or DVD drive.

Note that the filesystem types for the optical drive are specified as `udf,iso9660`, while the filesystem type for the floppy is specified as `auto`. For the optical drive, the mount process will check first for a udf filesystem (common on DVD) and then for an iso9660 filesystem (common on CD). For the floppy drive, the mount process will probe for a filesystem type. You can create or edit /etc/filesystems to change the order in which the filesystems will be probed.

Note: You should **always** unmount removable drives or media before disconnecting the drive or attempting to remove the media. Failure to do so may result in loss of data that has not yet been written to the device.

If you run a graphical desktop such as Nautilus, you will usually find options that allow removable devices and media to be automatically mounted. For example, if I insert a Knoppix DVD into the

DVD drive of my system, I might see a mount entry such as shown in Listing 12. The presence of 'uid=1000' shows that the user with id 1000 can unmount this disc. The `id` command shows the uid for user ian is 1000, so ian can unmount this disc.

Listing 12. Desktop mounting of DVD

```
[ian@echidna ~]$ mount | grep sr0
/dev/sr0 on /media/KNOPPIX type iso9660 (ro,nosuid,nodev,uhelper=udisks,
uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500)
[ian@echidna ~]$ id ian
uid=1000(ian) gid=1000(ian) groups=1000(ian)
```

You may also use the `eject` command to eject removable media when the drive supports the operation as most CD and DVD drives do. If you have not unmounted the device first, then `eject` will both unmount and eject the disc.

Swap space

You may have noticed in the discussion of `fstab` above that swap space does not have a mount point. The boot process usually enables swap space defined in /etc/fstab unless the `noauto` option is specified. To manually control swap space on a running system—for example, if you added a new swap partition—use the `swapon` and `swapoff` commands. See the man pages for details.

You can view the currently enabled swap devices with `cat /proc/swaps` or with `swapon -s` as shown in Listing 13.

Listing 13. Displaying swap space

```
[ian@echidna ~]$ swapon -s
Filename    Type  Size Used Priority
/dev/sdb1                               partition 514044 0 -1
/dev/sdb5                               partition 4192928 0 -2
[ian@echidna ~]$ cat /proc/swaps
Filename    Type  Size Used Priority
/dev/sdb1                               partition 514044 0 -1
/dev/sdb5                               partition 4192928 0 -2
```

This completes your introduction to device mounting on Linux.

Resources

Learn

- Develop and deploy your next app on the IBM Bluemix cloud platform.
- Use the developerWorks roadmap for LPIC-1 to find the developerWorks articles to help you study for LPIC-1 certification based on the April 2009 objectives.
- At the LPIC Program site, find detailed objectives, task lists, and sample questions for the three levels of the Linux Professional Institute's Linux system administration certification. In particular, see their April 2009 objectives for LPI exam 101 and LPI exam 102. Always refer to the LPIC Program site for the latest objectives.
- Review the entire LPI exam prep series on developerWorks to learn Linux fundamentals and prepare for system administrator certification based on earlier LPI exam objectives prior to April 2009.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

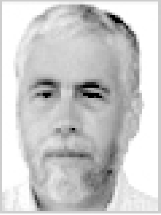
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Ian Shields



Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. Learn more about Ian in Ian's profile on developerWorks Community.