8.5.1 File System Mounting

Click one of the buttons to take you to that part of the video.

File System Mounting 0:00-0:33

One of the key differences between Linux and other operating systems, like Windows, is the fact that you need to mount a file system on Linux before you can use it. This is one of those things that really confuses new Linux system administrators.

Understand that on Linux you have to mount hard disk partitions, DVDs, USB drives, etc., before you can actually use them. The good news is the process of mounting a file system on Linux is really pretty easy. It's done using the mount command.

mount 0:34-5:27

The mount utility mounts the file system that has already been created on a partition or on an optical disk or on a flash drive, into an existing directory within the Linux file system. That way, when you switch into that directory, what you're actually doing is switching to the file system on that storage device.

To use the mount command, you enter mount, and then you use the -t option to specify which file system you want to use. Then you point to the partition that you want to mount-- and remember that this has to match what is configured on this. You can't just use whatever file system you want.

If I've created an ext4 file system on sdb1, then that's the type of file system I have to specify with the mount command.

Then, after the device file that we're mounting, we specify where in the file system we want to mount it at. In this case, I'm mounting it in a directory called /mnt/shared. You can see that in my /mnt directory, I already have a directory named /shared. That way, when I switch to this /shared directory, what I'm really doing is switching to this partition, the first partition, on the second hard disk in the system.

Here's a little tip. There may be times when you need to mount a storage device, say, a flash drive. And because it's a flash drive, you're not the one that actually partitioned and formatted it, and so you may not know what file system it uses. In these situations, you can use the -a option instead of the -t.

If you use -a, you don't have to specify a file system type. You still specify the device filename and the directory you want to mount it in. Using the -a will cause the mount command to try to mount whatever partition you specified, using all supported file system types until one actually works. And as soon as that's done, it goes ahead and completes the mounting operation.

Be aware that on most distributions, the /mnt directory is the default directory that's used for mounting both local and remote file systems. When I say local, I'm talking about locally connected devices, such as a USB Drive or an optical disk, while a remote file system is one that's being accessed on another computer over a network connection.

You don't have to use /mnt to mount a file system if you don't want to. You can mount a partition into any directory in the Linux file system that you want. However, by convention, /mnt is used by default for this purpose.

On most distributions, /mnt is used to mount not only partitions on a hard drive, but also external flash storage devices, like a USB drive. It's also used for mounting optical devices, such as a DVD. However, be aware that some distributions will kind of split things up. And they will use /mnt for mounting partitions on hard drives and they will use the /media directory for mounting removable media, like an optical disk or a USB flash drive.

Be aware that there are other options you can use with the mount command that we haven't covered here. Go ahead and look at the man page for mount. You can see a complete listing of all the available options.

For example, you can use the -o to specify mounting options that are used as that device is mounted into the file system. A good example would be -oro, which mounts whatever device you specify into the directory you specify, but mounts it read-only.

You can read information from it, but you can't write information to it. After mounting the partition, you can then use the mount command with no options and no parameters to simply view all of the mounted file systems.

This is a very useful command that you should remember. You can then look through the output to verify that the device you specified is actually mounted in the directory you specified. In this case, we mounted /dev/sdb1 in the /mnt/shared directory, using the ext4 file system.

You need to be familiar with this file right here: /etc/mtab. This is an important file. Understand that whenever you mount or unmount a device in the Linux file system, this file gets updated. Therefore, it always contains the most current list of currently mounted file systems, and it is just a text file. Therefore, you can view it using a standard text-viewing utility, like cat, or less, or more, and by doing so, you can see a list of mounted file systems.

And here you can see that we have sdb1 mounted in the directory we specified, with the file system we specified. In addition to the /etc/mtab file, you can also view the contents of this file to view information about mounted file systems: /proc/mounts.

Again, it's just a text file so you can use cat, less, or more-- whatever it is you want to use to view it. Again, we see the partition that we mounted in this directory using the ext4 file system.

---

umount 5:28-7:20

In addition to mounting a file system, you can also unmount a file system if you need to. This is done using the umount command. As soon as I say this in every single class, a student pipes up and says, "You spelled that wrong. It's not umount, it's unmount." No, actually, it is not unmount. It is truly umount.

To unmount a file system, you enter the 'umount' command, followed by one of two different things. Either the device that you want to unmount such as /dev/sdb1, or you can also specify the directory where that device is currently mounted. In this case, /mnt/shared. Either one of these commands would accomplish the exact same thing.

It's important that you remember that when you're trying to unmount a file system, the device that you're trying to unmount cannot be busy. If it's busy, then it's not going to unmount. For example, in this case I have used the cd command to change into the /shared directory, which is where this device is mounted.

And then I tried to use the umount command to unmount this device and it says, "I can't, I'm busy. Somebody is using it." The same thing would hold true if I had mounted an optical disk or a USB drive somewhere in my file system and I were using the file system on that device.

Well, if I tried to unmount it with the umount command, it will give me a similar error saying, "Hey, somebody's using it. I can't unmount it right now." This brings up another issue when you're working with Linux file systems. Understand that just because you mount a file system with the mount command does not mean it's going to stay mounted.

As long as the system stays running, yeah, it will stay mounted, but if you reboot the system, the device that you specified with the mount command will not be remounted by default when the system starts back up. But you can fix this.

Syntax of the /etc/fstab 7:21-12:33

What you do is go into your /etc/fstab file and then add the mounting information for that device to this file. That way, when the system boots up, you can be sure that that device will be mounted in the directory you specify. I did that right here. I've added the /dev/sdb1 device to the fstab file.

I specified where I want it mounted at, what file system I want to use, and then I specified my mount option here. We'll go into that in a little more detail here. The important thing for you to remember is the fact that each line in the fstab file contains six different fields that you need to fill out for each file system to be mounted whenever the system boots.

The first field specifies the device to be mounted. In this case, /dev/sdb1. The second field specifies the mount point where you want that device to be mounted; in this case, we want it mounted in mnt/shared. The next field specifies the file system that is being created on that device. Remember, that does have to match. You can't just use whatever file system you want.

The file system you specify must be the file system that has been, already been, made on that particular device. We have our mount options, which we'll talk more about in just a second. I put defaults in here, which will just use the default mounting options. You can customize this entry to make this device be mounted into the file system, using a wide variety of different options.

The next field specifies whether or not this file system should be dumped. 0 means don't dump it. 1 means go ahead and dump it. Then the last number specifies the order in which the file system check utility should check the file system when the system starts up. Basically, there are only two different values that are used in this field.

The file system that contains the root partition should always be checked first. On that one line in the fstab file, this field would be set to 1. Everything else will be set to a value of 2, meaning it really doesn't matter. Basically, we want the root file system always checked first, and then after that, it can be checked in whatever order we want.

In that fourth field, we talked about the fact that you could use a variety of different mount options. I just used defaults just to make it easy, but there are many different mount options you could include in this field.

You can use the mount options shown here, and you can use many of them all at the same time if you want to. All you have to do is put a comma between the different mount options that you want to use.

rw specifies that we mount the file system read/write. We can read data from it and write data to it, whereas ro allows us only to read information. It does not allow us to write information.

The 'sync' option enables synchronous input and output, which basically means that any changes you make to a file on that file system need to be written immediately. This is commonly used for devices that are removable in nature, such as a USB drive. Because there's a potential lag between the time a write operation happens and the time when somebody pulls the drive out of the USB port, you probably want your USB drive information to be written right away and not held.

The opposite is the async option, which enables asynchronous input/output. This option is used typically on devices that will not be disconnected from the system at all, like a hard disk drive. By enabling async, all the changes that are being made to the files in the file system may be cached and then they will be written later on, when the hard disk isn't busy.

By doing this, we increase performance. But the gotcha here is that drive has got to be in the system all the time; otherwise, you could potentially miss a write operation. The atime option specifies that whenever a file is accessed, the time when it's accessed will be updated in the file's inode.

The noatime does just the opposite. It does not record the last access time in the file's inode. Sometimes that is used because it does provide a better overall file system performance because we're not constantly updating the inodes of the files with their last access time.

nodev prevents the device files in /dev from being interpreted as special block devices, whereas dev just does the opposite. It causes the kernel to interpret device files in /dev as special block devices. We also have the noexec option, which prevents binaries on that file system from being executed. This can be useful from a security standpoint. And the exec does just the opposite. It allows any executables on the file system to be executed.

The nosuid option blocks the use of suid and sgid permissions. Where suid does just the opposite. It enables the use of suid and sgid permissions.

auto specifies that the file system be automatically mounted whenever the system boots. noauto does just the opposite; it prevents the file system from being automatically mounted on system boot.

The user option allows users to mount the file system, while nouser specifies that only root is allowed to mount the file system.

And as we said earlier, you can just use defaults if you want to, which if you specify, will cause the kernel to mount the file system using the read/write option, the suid option, the dev option, the exec option, the auto option, the nouser option, and the async option.

ISO Image Mount 12:34-13:34

Before we end, there's one more thing I want to show you, and that's how to mount an image file-- an ISO file, in other words-- into the Linux file system. By doing this, we essentially mount the ISO image into a directory. When we switch to that directory, we're actually able to access the contents of the ISO file.

The syntax for doing that is shown here. First, we run mount, and then we use the -o to specify our mount options. And we specify the loop option with -o. This specifies that we're mounting a loopback device. Then, we specify the filename of the image file, whatever that happens to be, and then the mount point where we're mounting it into the file system.

An example is shown here, mount -o loop myfiles.iso into /mnt/iso. This can be really useful because it allows you to take an ISO file in your file system, mount it into a directory somewhere, and then access the contents of the ISO file-- basically, as if it were an optical disk in an optical drive, such as a DVD in your optical drive.

Summary 13:35-13:49

That's it for this lesson. In this lesson, we discussed how you mount Linux file systems. First, we looked at the mount command along with the umount command. Then we looked at the syntax of the /etc/fstab file. Then we ended this lesson by talking about how you can mount an ISO image in the Linux file system.
