8.4.1 Linux File System Types

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

Linux File System Types 0:00-0:46

In this lesson, we're going to review several commonly used Linux file system types. Understand that whenever the Linux kernel needs to conduct an I/O Operation--an input operation or an output operation--to one of your system's storage devices, the kernel needs to know where that data is stored, how it can access it, and also where it's safe to write other new information.

This is the job of the file system. The file system reliably stores data on the storage device and organizes it in such a way that it can be found easily. When you use a file browser, or maybe a cd command at the command prompt to navigate through directories within the file system and then open up a file, it's the file system that makes this entire process possible.

---

ext2 0:47-4:34

Let's begin by talking about the ext2 file system. ext2 stands for Second Extended File System, and it is probably one of the oldest Linux file systems that's still available. It was introduced a long time ago--in 1993 I believe.

It stores data in the standard hierarchical fashion that is used by most other file systems. Data is stored in files, files are stored in directories, and a given directory can contain either files or other directories, which we call subdirectories.

The maximum file size for a file on an ext2 file system is 2 terabytes (TB), and the maximum size for an ext2 volume is 4 TB. It supports filenames that are up to 255 characters long, and it supports our Linux permissions, and it also supports file compression.

The ext2 file system is a fantastic file system. It's been around for a long, long time. Long enough for most of its bugs to be worked out. In fact, I daresay it's probably the most widely used Linux file system that's ever been implemented, just because it's been around for so long.

However, you need to be aware of the fact that ext2 has one key weakness that has led to the development of other Linux file systems.

This is the fact that the ext2 file system takes a very long time to recover if the system shuts down abruptly.

When shutting down the Linux system, the operating system needs to first cleanly dismount the file system. This ensures that all pending file system transactions are actually committed to disk before the system shuts off. The problem arises when the system shuts down without completing this clean dismount procedure.

Let's suppose we have a power outage and the Linux system shuts off suddenly without going through the proper shutdown procedure. When this happens, it's possible that there were pending disk transactions waiting that weren't actually completed.

In order to keep this from happening, you do need to use a UPS. You're using one right? If not, I suggest that you do so. I lost data once on an ext2 file system due to a power outage and it was a bad thing because I didn't have a UPS hooked up. I've sworn to never ever, ever, let that happen again.

If this happens, in order to clean up the file system afterwards, the ext2 file system will automatically run a program called e2fsck, which stands for ext2 file system check. It will run this program the next time the system starts up.

This utility will try to fix any problems that were created when the system went down without properly dismounting the hard disk drives first. If it finds any files that are unallocated, or unclaimed blocks of data, it will take that information and it will write it to a special directory you'll see in the file system called lost+found.

By doing this, ext2 tries to ensure that the data integrity is maintained in the event of an improper shutdown.

This all sounds good, right? But it's still got a problem and that is the fact that the e2fsck utility will analyze the entire file system when this happens, not just the few files that were in the process of being modified when the system went down uncleanly.

On a very basic, minimal Linux system, this can take a while--10 to 15 minutes. But on an extensive system, like a Linux server that has lots of file system data, this process can take hours. If you think about it from the system administrators perspective, it's bad enough that the system went down unexpectedly in the first place.

Now you've got to wait hours for the system to get back up and running again. Your end users and your managers are not going to be happy.

Because of this very issue, other Linux file systems have been developed and are pretty much completely replacing ext2, although you still might run into an ext2 system here and there.

ext3 4:35-7:47

The first of these is ext3. Really, ext3, as you can see by its name, is just an updated version of ext2. ext3 stands for Third Extended File System. In fact, ext2 and ext3 are so similar that most of the file system utilities used by ext2 are also used by ext3.

In fact, you can easily upgrade an ext2 file system to ext3. Or, you can go the other direction if you wanted to. You can downgrade an ext3 file system back to ext2.

You probably would never want to do this, because the ext3 file systems offers one key advantage that makes it highly preferable over ext2. That is the fact that it uses journaling.

Remember that the key disadvantage of ext2 is the fact that it had to check the entire file system if the system went down uncleanly? Journaling eliminates this problem. The journaling process is shown here. Before committing a transaction to the hard disk drive, such as saving a file, the ext3 file system will record that transaction to the journal and mark it as pending or incomplete.

In this case, we need to write data to a file named schedule.odt. Then after that process is actually complete--after that save process, in this case to schedule.odt, is done--then the ext3 file system marks the transaction as completed.

In essence, the journal says, "Here's what I'm going to do," and tracks whether or not that actually got done. By doing this, the ext3 file system can keep a log of the most recent file transactions and whether or not they were actually completed.

Using a journal, if an event occurs, like a power outage, that causes the system to go down uncleanly without properly dismounting the disk, then the ext3 file system will replay that journal when the system comes back up. This allows the file system to verify the data on the disk and bring it back into a consistent state, if possible, using the information stored in the journal.

Unlike ext2, the ext3 file system doesn't need to check the entire file system. This is the key benefit of ext3 over ext2. Because ext3 maintains a log of the most recent transactions within the journal, the ext3 file system simply has to check the transactions within the journal that are listed as incomplete.

Therefore, using journaling, the disk recovery time after an improper shutdown takes dramatically less time than what we saw using ext2. Instead of taking hours, the ext3 file system can replay the journal in only a few seconds, maybe a few minutes if necessary--even if the file system is very large--and the system is back up and running again.

The disadvantage of ext3 is the fact that the journaling process does use up more system memory and it does slow down the disk I/O process slightly. However, because it does such a better job of ensuring data integrity and it does it faster, most system administrators prefer ext3 by far and away over ext2, even though it does slightly decrease overall disk performance.

Reiser 7:48-8:31

Let's look at the Reiser file system. The Reiser file system is an alternative to the ext3 file system, and like ext3, Reiser uses journaling to make crash recovery fast. However, the two are not compatible. Reiser is a completely different file system from ext2 and ext3.

It uses a completely different internal structure. This internal structure is optimized to allow bigger file sizes and bigger volume sizes. As you can see, Reiser allows a maximum file size of 8 TB and a maximum volume size of 16 TB, and it's reputed to be faster than ext2 or ext3.

---

ext4 8:32-9:49

You should also be familiar with the ext4 file system. ext4 was released in late 2008, and as you might guess from its name, fourth extended file system. It's really just an updated version of ext3. Just as ext3 is backward compatible with ext2, ext4 is backward compatible with ext3--and ext2 for that matter.

The ext4 file system supports volume sizes up to 1 exabyte (EB) in size and files can be up to 16 TB in size. You can have a maximum of 4 billion files in an ext4 file system. And as with ext2 and ext3, the maximum filename length is 255 characters.

Just as with ext3, ext4 also uses a journal. ext4 incorporates additional functionality with the journal itself. ext4 uses checksums to verify the integrity of the journal file. This helps improve the overall reliability of the system, because if you think about it, that journal file is probably the most heavily used file on the hard disk drive, and if it gets corrupted you've got real problems. Using checksums helps ensure that that file does not get corrupted.

---

btrfs 9:50-13:38

With this in mind, let's shift gears and look at the btr file system, or btrfs. btrfs is a newer Linux file system that really is totally and radically different from the other file systems that we talked about previously in this lesson.

The btrfs uses a newer type of storage technology called copy-on-write. It really allows us to create a file system on Linux that is very similar to the NSS file system that's found on Novell Products, as well as the storage space technology that's found on later versions of Windows.

Using copy-on-write technology, btrfs allows us to implement some features that are not found in the earlier Linux file systems that we just talked about. Specifically, storage pools and storage snapshots.

Really, the btrfs file system provides an alternative to the traditional process of creating disk partitions and installing a file system on those partitions. With btrfs, what you do instead is create a storage pool composed of all the storage space on the storage devices on your system.

If you're looking at this saying, "That looks like LVM." You're right. It's very similar to LVM, but it includes features that LVM does not include. Once we've allocated our storage devices to a storage pool, we then can define storage volumes from the storage pool, and then we can mount those storage volumes at various mount points in our Linux file system.

This system provides a great deal of flexibility with the storage space on your system. For example, let's suppose we've defined a storage volume that's mounted in /home, which is where all of our user's /home directories reside, and where they store all of their video files that they've downloaded from the internet.

It's very common to run out of space in /home. Using btrfs, if we want to add capacity, all we have to do is install a new hard drive in the system, then allocate its space to the pool where the volume mounted in /home lives.

When you do, the size of the volume is automatically increased. You didn't have to back up any data, you didn't have to resize partitions, you didn't have to restore data as you would have had to do if you were working with traditional partitions.

In addition to storage volumes, btrfs also provides snapshot functionality. Snapshots do a great job of protecting your data. Essentially, what snapshots do is take pictures, if you will, of what your data looked like at specific intervals and saves it on a separate media.

For example, we have snapshot 1 snapshot 2, and snapshot 3 of storage volume 1. This way, if a file ever gets lost from the storage volume or gets corrupted for some reason, we can come over here to the appropriate snapshot file--say, snapshot number 2--grab the file that we need to restore, and drop it back into the file system on the storage volume.

Basically, it allows us to restore that lost file in just a matter of seconds instead of having to restore it from tape or some other backup medium.

So which file system should you use? Well, it depends upon your personal preferences and the needs of your particular Linux deployment. My choice for many years was the Reiser file system. However, lately ext4 has been my file system of choice, but I'm really liking the looks of the newer btrfs.

There are many other file systems that you could use with Linux if you wanted to. For example, you can use the XFS file system from Silicon Graphics, or you can even use the VFAT or NTFS file system from Windows. Although, be aware that NTFS support is there but it's not 100% baked yet, and you're better off using a standard Linux file system.

Summary 13:39-13:49

That's it for this lesson. In this lesson we discussed the Linux file system types that you can use. We looked at ext2, we looked at ext3, Reiser, ext4, and then we ended this lesson by talking about the btr file system.
