8.6.2 Maintaining File Systems

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

Maintain File Systems 0:00-0:23

In this demonstration, we're going to discuss maintaining Linux file systems. We'll first look at several utilities you can use to check disk space usage. We'll talk about how to check for open files, we'll talk about how to identify processes that have files open in the file system, and then we'll end this demonstration by talking about how to check the integrity of various file systems.

Check Disk Space Utilization 0:24-3:46

Let's begin by talking about how to check disk space usage. First thing I need to do before and I can do anything is switch to my root user account, because I will need root level privileges to do the things we're going to do.

One of the utilities that you can use to check disk space usage is the 'df' command. df is a very useful utility, but be aware that, by default, it displays the output in a format that can be somewhat difficult to read and might require you to do a little bit of math to determine the actual space consumed on a disk, as well as the space currently available.

If, however, you run df with a '-h' option, it will reformat the output in a human-readable form, which will reformat all of our space data in megabytes and gigabytes, making it much easier to read and interpret. When we run df, we see a couple of key pieces of information.

First, we see a list of all the partitions in the system, and we see where each of these is mounted in the Linux file system. We see the total size of the partition, we see how much is used, how much is available, and then a percentage figure over here which is calculated from these two columns.

For example, here we have two partitions that we have created, /dev/sdb1 and /dev/sdb5. sdb1 is mounted in the /mount/shared directory, while sdb5 is mounted in the /mount/private directory. The size of sdb1 is 9.8 GB, roughly 10 GB in size, while sdb5 is about 5 GB in size. sdb1 has 23 MB of space consumed.

There's still 9.2 GB available on that disk, so we've really only used about 1% of the available space in that file system. sdb5 is similar.

It has a little bit more space consumed--39 MB--but it still has 4.6 GB available, so we still have consumed only about 1% of the available space on that drive.

df -h is a tool I use on an almost daily basis to make sure that my disks don't run out of space. There's a similar utility that you can use also to monitor disk space usage, and it's called the du command. You commonly see Linux administrators use df and du together as they monitor disk space utilization.

du is particularly useful in situations where you have a multiuser Linux system where you have lots of different users all having home directories on the Linux system and saving files there. Let's switch to the /home directory.

If we do an 'ls' command, we see that there is only one user assigned to this system, but if I had 10 users on this system, I would see 10 home directories in /home. From the /home directory, we can enter 'du -h' and when I do, I can see how much space is being used by each of the subfolders within /home.

In this case, it's just going through the rtracy directory and looking at each of my subdirectories within rtracy and specifying how much space is used by each of those directories. As you can see, I don't have a whole lot of space being consumed at this time.

If you've got an end user who is downloading huge amounts of data off of the web and saving it on your server and they're chewing up all of your available disk space, requiring you to go out and buy extra hard drives all the time, you can use this utility to find out who that is.

Look For Open Files 3:47-5:45

Another useful utility is the lsof command. The lsof command checks for open files on the system. You can see that we use the ls command normally to just generate a listing of files and folders in a particular directory, so we basically are generating a list of open files. Its name identifies kind of what it does.

This command is very useful in situations where you have, say, a Linux server and you have users who are using their client systems to connect over the network to the Linux system and access files on that server. If their remote system, their workstation, crashes for some

reason while they've had a document open, say a word processing document, then they're not going to be able to open that file again until it's manually closed.

You can use lsof to troubleshoot the problem. You can use it to find the files that got left open when their system crashed, providing you with the information you need to get those files closed so that they can access their data again.

For example, let's run 'lsof -s -u' and then let's enter 'rtracy'. When we use the -s option with the lsof command, all of the open files will be displayed and their sizes. That's what -s does.

A very useful option and the one I use all the time is -u. -u configures lsof to look for just the open files that are owned by a particular user. In this case, I'm just going to look for files that are opened by the rtracy user.

This can be really useful because if you don't use the -u option, then lsof is going to show all open files that are owned by all users and you're going to get an avalanche of data. It's overkill and it's going to be very difficult to find the data that you need.

By constraining it with the -u option, we can just see the files that were opened by a particular user. Let's go ahead and run the command. When we do, we see a list of all the files over here that I currently have open as my rtracy user. We can see that my rtracy does have quite a few files currently open.

Find Processes That Have Open Files 5:46-6:48

A related command that you can use is called the fuser command. fuser identifies processes that are currently accessing files in the file system.

In other words, it looks for processes that have opened some file in the file system and have not closed that file yet. The file is in an open state. To do this, we run fuser and one of the useful options with fuser is to specify a particular directory with the -m command.

This helps you constrain the output of fuser to just the data you need to see. For example, you could use the fuser command to identify a process that has a particular file open in an end user's home directory. We can enter 'fuser -m' and then specify the directory in question, /home, and this will display the process ID numbers--all the processes that are using a particular file in the /home directory.

Go ahead and press Enter. In the output of the command, we see a list of all the different processes running on the system that have a file open in /home.

Check the Integrity of File Systems 6:49-11:12

The last thing we're going to talk about in this demonstration is how to check the integrity of your Linux file system, because it is possible for files and directories within the file system to become corrupt. To check the integrity of file systems, we use the fsck command: file system check. We'll go ahead and clear the screen here so we can see what we're doing.

The fsck command checks the integrity of both files and directories within the file system. Be aware that this utility is actually automatically run at system boot if Linux detects that it was previously shut down uncleanly. Maybe we had a power outage or we don't have the system hooked to a UPS like we should, and therefore the system went down uncleanly.

The situation will be detected and file system check will be automatically run on all the mounted file systems the next time the system boots to make sure that none of the files were damaged when the system went down uncleanly.

Even though it is run automatically in this situation, you can also run it manually any time you need to. I do need to point out too that the fsck command is actually just a front end. There are separate fsck utilities for each Linux file system type.

Let's do a 'find' command and start at the root of the file system and look for any file that begins with 'fsck.' When we do, you can see that there's actually an fsck utility for the .ext2 file system, for the .ext3 file system, .msdos, .ntfs, .fat, .minix, .vfat, .xfs, .ext4, and so on.

Before you can check a file system with the fsck utility, you first have to unmount it. If I run the 'mount' command here, we will see that /dev/sdb1 is currently mounted in the /mount/share directory and sdb5 is mounted in the /mount/private directory. We want to check these file systems so we first must use the 'umount' command, not unmount, umount to unmount the '/dev/sdb1' file system and also the 'sdb5' file system.

We can run the fsck command on both of these file systems. A file system check will automatically look and see what type of file system has been created on these partitions and run the appropriate utility. For example, I believe sdb1 has been formatted with the ext4 file system and sdb5 has been formatted with the ext3 file system.

We don't need to worry about that. The file system check will just take care of it for us. To run it, we enter 'fsck /dev/sdb1' to check the first file system and there aren't actually very many files in that file system, so it took only just a few seconds to actually run.

The key thing we want to look for is this output right here where it tells us that everything is clean, no errors were found. Let's go ahead and do the same thing on the sdb5 partition, and notice that once again the file system is clean. We're good to go.

At this point, we would want to then remount these two file systems back in the file system so that we can access the data that's on them. One way to do this would be to use the mount command, and that would work just fine, but it takes a little bit of typing. But because both of these file systems have the appropriate entries in the fstab file--/etc/fstab--we can actually just use the '-a' option with the 'mount' command to tell mount to go ahead and go through the /etc/fstab file, identify any of the file systems listed here that are not currently mounted, and mount them automatically for us.

Enter, and now if we run 'mount', we should see that they have both been remounted in the appropriate directories. Just a quick shortcut.

Before we end, there is one other thing I need to tell you about fsck. That is the fact that if you run 'fsck' without specifying a particular file system to look at, such as /dev/sdb1, then what fsck is going to do is go through your /etc/fstab file and will automatically check all of the file systems that it finds in here. Again, they do have to be unmounted first.

Summary 11:13-11:31

That's it for this demonstration. In this demo, we talked about maintaining the Linux file system. We first looked at utilities you can use to check for disk space utilization. We then talked about some utilities you can use to look for open files. We talked about utilities you can use to find processes that have files open in the file system, and then we ended this demonstration by talking about how to check the integrity of file systems.
