2.10.1 Links

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

Links 0:00-0:23

The Linux file system supports a special file type called a link file. Link files don't actually contain any content in the way that a regular file does. Instead, they are redirectors that point you to a different file or directory in the file system. Now, in Linux, you can create two different types of link files.

Let's take a look at what they are.

Hard Link Files 0:24-2:06

The first type of link file that you can create on a Linux system is called a hard link file. A hard link is a file that points directly to the inode of another file. Now, remember, the inode for a file stores basic information about the file in the Linux file system, such as its size, what device it's on, who owns that file, and the permissions assigned to that file.

Essentially, with a hard link file, we have two different files that share the same inode number, and this is important to understand. Because these two files use the same inode, you actually can't tell which file is the pointer and which file is the pointee after the hard link is created. Basically, it's as if the two files were the same file, even though they actually exist in different locations in the file system.

In this example, I have two separate files in the file system. One is /home/rtracy/myfile. The other is /opt/myfile. We've taken both of these files and find them as a hard link. Therefore, they share the same inode number. Therefore, as far as the Linux system is concerned, this file, right here, is exactly the same as this file, right here. In fact, if I were to open up this file in, say, a text editor, and insert some data in it, if I were then to come over and look at the contents of this file, say, with the cat utility, I would see exactly the same contents as I put in this file because, again, as far as the Linux file system is concerned, they're the same file, even though they are separated into different directories.

Symbolic Link Files 2:07-3:29

In addition to hard links, you can also create symbolic links. A symbolic link file also points to another file somewhere else in the file system. However, a symbolic link has its own inode, unlike a hard link. Because the pointer file has its own inode, the pointer and the pointee in the file system can be easily identified. Essentially, a symbolic link is much more like a traditional shortcut file. The two files stay unique from each other even though one does point to the other.

Once again, we have the /home/rtracy/myfile file. We also have the /opt/myfile file. In this case, we are creating symbolic link between the two, where my file is the pointer, and myfile in /opt is the pointee, which basically means that this file, here, points to this file, over here. Because this is a symbolic link instead of a hard link, this file has its own inode number, and this file has its own inode. Essentially, they are two separate files, even though this file, here, points to this file, here. If I were to go into a text editor and open up this file and write data to it, it would be written to this file as well, even though they are still, physically, two separate files.

View Symbolic Links: ls Command 3:30-4:10

You can view symbolic link files as well as the files they point to using the ls -l command. Notice that we had to use the -l option here, with the command. It does not work if you just type —˜ls'. If you type 'ls -l' in a directory where a symbolic link resides, you will first see the name of the pointer file, the link file, and then a little arrow, and then the name of the file that the link file points to, the pointee file. You can see in this example that we have a file in my user's home directory named logfile, and it points to the /var/log/boot.log file.

Create Links: ln Command 4:11-5:49

If you want to create a link file, you can do it using the ln command. An example of how to do that is shown here. We type 'ln'. Then we specify the pointee file, the file that's being pointed to, and the pointer file, the file that's going to do the pointing. In this case, we are creating a link file named docs that points to /user/share/doc.

Now, it's important to note that, in this case, we added a -s option. If you do not use the -s option, then the link will be created as a hard link, where the two files share the same inode. With the -s option, it tells ln not to create a hard link, but to, instead, create a symbolic link.

Now, before we go on, I want you to pay careful attention to the syntax because it can be a little confusing. It actually does trip up a lot of new Linux system administrators because we specify the pointee file, right here, first, not the pointer. You know, common sense would have us think that we specify the pointer file and then the file that the link file points to, the pointee, but it's not. It's just the opposite. We specify the file that's being pointed to first, and then the file that does the pointing last.

In this example, you can see that I type the ls -l command after running the ln command, and we see that the link file has been created. We now have a link in our user's home directory named docs that points to /user/share/doc. Of the two types of link files, you'll probably spend most of your time working with symbolic links versus hard links.

Summary 5:50-6:01

That's it for this lesson. In this lesson, we talked about how link files work. We first talked about the two different types of link files, hard links and symbolic links. Then we talked about how to view link files with the ls command. And then we talked about how to create link files using the ln command.
