2.10.2 Create Links

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

Creating Links 0:00-0:15

In this demonstration, we're going to look at managing link files. Specifically, we're going to talk about how to view link files. We're going to talk about how to create symbolic links. And then we'll end this demonstration by talking about how to create hard links.

View Link Files 0:16-1:56

Let's talk about our first topic, viewing link files. In order to do this, I'm first going to switch to a directory in the Linux file system where I know a lot of link files already exist. They're created by default. That is the /dev directory. I'm going to enter 'cd' for change directory, and we're going to change to /dev.

You can view link files, as well as the directory that they point to, using the ls command. The ls command by itself isn't enough. It won't show you the target where a link file points to by itself. In order to do that, we need to include the -l option, which formats the output of the ls command in long format. If we scroll up here a little ways, we can see the listing of various files in the /dev directory is displayed by the ls command. Any file that is a symbolic link, however, will have this character, right here, added, which tells you where that link file points to. For example, right here, we have the cdrom file, which is a symbolic link, and it points to another file within the same local directory called sr0. In fact, if we scroll down here, we can probably find the sr0 file right there. So, the cdrom file points to this file right here, sr0.

There are other symbolic link files in the same directory as well. For example, the core file, right here, is a symbolic link, and it points to a file somewhere else in the file system--in this case, /proc/kcore. That's how you view link files; you use the ls command with the -l option. With that in mind, let's create some links.

Create Symbolic Link File With ln 1:57-3:33

We'll first create a symbolic link file. I want to create that symbolic link file in my /home directory, so I will enter 'cd ~' at the shell prompt to switch to my /home directory. Let's suppose that we want to create a symbolic link in my /home directory that points to the documentation directory on the system so I have quick access to the documentation files. The directory that I want to point to is the /usr/share/doc directory. If I do an 'ls' here, I can see there are lots of subdirectories within this directory that contain documentation for different Linux system components. Go back to my /home directory.

To create this symbolic link, I type 'ln -s' to create a symbolic link. If you leave the -s option off, it creates a hard link, and that's not what we want to do in this case. After the -s option, I need to specify where the link will point to. This can be a little confusing, at least for me, because in my mind, I would want to specify the symbolic link file first and then the file or directory that it points to second. That makes sense to me, but that's not how the ln command works. Instead, we specify the target first, and then we specify the name of the link file that we want to create. The target we want to point to in this case is /usr/share/doc. Now we can specify the name of the symbolic link itself that's going to be created in the local directory; let's just call it docs. In this example, the symbolic link will be named docs, and it will point to /usr/share/doc.

View Created Symbolic Link 3:34-4:44

Now if we use the ls -l command in my /home directory, we should see the docs folder has been created, and it points to /usr/share/doc. As you can see, in this distribution, the symbolic link files are colored light blue; that will not be the case across all distributions, but it is fairly common. So, if you see a light blue file, it's very likely that it is a symbolic link that points to another file or directory in the file system. With this symbolic link created, I can now enter 'cd docs' from within my /home directory. And now, if I type the ls command, I see all the files and directories that are located in /usr/share/doc, even though, according to the Linux shell, I'm still in my /home directory. If I do the pwd command, it tells me I'm in /home/rtracy/docs. But the files and directories I'm seeing up here are in /usr/share/doc.

Create a Hard Link File 4:45-6:00

Let's change back to my /home directory. And now, let's create a different type of link file. This time, we want to create a hard link file. So, the first thing I need to do is actually create the link file that I'm going to point to. You can create this in any way that you want. For our purposes, I just want to create a text file, so I'm just going to use the touch command to create a new empty blank file, and we'll call it

pointee. Do an ls command; now we should see the pointee file. This is just a blank empty file that we're going to create a hard link that points to. To do this, I enter 'ln' without the -s parameter, and then I specify the name of the target of the link, just like we did before, which will be the pointee file that we have right here. And then I specify the name of the pointer file. And, for sake of simplicity, let's call it pointer.

In this case, pointer will point to pointee. But because we're not creating a symbolic link this time--we're creating a hard link--things are going to be a little bit different. Press Enter. If I type the ls command, we see that

---

View a Hard Link 6:01-6:42

we have the pointer file that points to the pointee file. But notice that they're not colored like the symbolic link is, over here, because it's a different type of file. In fact, if I run the ls -l command, notice that they do not have any pointer information associated with them like the symbolic link that we created earlier.

If we were to just look at these, we might initially think that these are just plain old files in the Linux file system, nothing special about them. But there is something special about them. As far as the Linux file system is concerned, basically, these are both the same file, and evidence of this can be seen using the -i option with the ls command.

View Inode Information 6:43-9:13

We'll enter ls -i, and we just want to see the files in the local directory that begin with a lowercase p. The -i option causes the ls command to display inode information. Press Enter. Notice something special here: both pointee and pointer have exactly the same inode number. As I said earlier, Linux will, essentially, view these two files as if they were the same file because they have exactly the same inode.

To demonstrate this fact, let's open up the pointer file right here, in a text editor. Enter 'vi pointer', and we'll press Insert, and we'll just add some text. Press Escape. We'll write our changes to disk by entering 'exit'. Now I can type 'cat pointer', and we see the text that we added to the file. The cat command simply displays the contents of a text file on the screen. Now let's do 'cat pointee'. And notice that they have exactly the same text; I added the text to the pointer file, and it appears in the pointee file because, again, as we said earlier, Linux views these two files as basically the same file because they have the same inode--that's what a hard link is versus a symbolic link. Symbolic links have independent inodes. To view this, let's run the ls command again with the -i option, and let's identify the inode of the docs symbolic file that we created earlier. It has an inode of 927576. Now let's run the ls -i command in the /usr/share directory. And if we scroll up here, we can identify the inode of the doc folder that that symbolic link points to. As you can see, it's quite different; it's got an inode of 270457.

With a symbolic link, the symbolic link file has an inode that is different than its target. So, they're basically two independent files--one just points to another. With a hard link, though, the inodes are exactly the same.

Summary 9:14-9:27

That's it for this demonstration. In this demo, we looked at how to view link files with the ls -l command. We then created a symbolic link file with the ln command. We then created a hard link file with the ln command. And then we talked about the differences between hard links and soft links using the ls -i command.
