2.9.4 Manage Files

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

Manage Files 0:00-0:17

In this demonstration we're going to talk about managing files in the Linux file system. Specifically, we're going to talk about creating files, copying files, moving files, deleting files, and getting information about files.

Create Files 0:18-1:40

Let's begin by talking about how to create a new file in the Linux file system. There are basically two options for doing this. One is to load an application, such as a text editor, and create a new file. Let's go ahead and do that here using the vi utility. vi is a text editor.

We're going to run vi and create a new file called 'mynewfile'. We'll go into insert mode and add some text: "This is my new file." We'll write our changes to the file and 'exit' the editor. We'll run the 'ls' command. We now see that we have a file called 'mynewfile' that we just created with vi. That's one option for creating a file.

There will be occasions when you just need to create a new file and you don't really care whether or not it has content in it yet or not. You might be planning on doing that later or planning on letting an application write information to that file, in which case you can just use the touch command to create a new file.

All you have to do is run 'touch', followed by the name of the file that you want to create. Let's call it 'mynewfile2' in this case. Let's do an 'ls -l' command. We can see that we have a file called mynewfile that was just created. It is 0 bytes in size because it's empty, it's blank, whereas the file we created with vi had a little bit of text in it, so it's 21 bytes in size.

Copy Files 1:41-4:20

You can use other shell commands to copy files to other locations in the Linux file system. This is done using the cp utility. As you might guess cp stands for copy. It's important to note that because the files are being copied with cp, the source file is going to be preserved and a new copy of that file is going to be created wherever you specify in the file system.

The syntax for using cp is to enter 'cp' first, and then you specify which file you want to copy. Let's copy the 'myfile' file right here in our Linux file system. Notice that I did not provide any path information for myfile. This is called a relative path.

Because we did not specify any path information, the Linux shell is going to assume that we want the myfile file is located in the current directory.

On the other hand, if you wanted to specify a file that resided somewhere else and not in the current directory, you would have to specify the full path to that file. For example, let's copy this file to a different directory in the file system.

Right now I am in my home directory, as you can tell with the ~, which is /home/rtracy. Let's copy 'myfile' to the '/tmp' directory. This is an explicit path, because we start at the root of the file system / and then we specify all the directories in between the root and the directory where we want to copy the file.

We're going only one layer deep into /tmp, but because we started at the root of the file system, this is an explicit path and this is a relative path. Let's go ahead and press Enter, the file is copied and now we can use the 'ls' command to look at the contents of the /tmp directory. You see that 'myfile' is there.

If we do an 'ls' of the local directory, we see that myfile is still here. Remember, copy leaves the source file intact and makes a new copy of that file somewhere else. It's important to note that you can actually rename a file during the copy process and it's pretty easy to do. All you have to do is run the 'cp' command specifying the source file.

Instead of specifying a source destination, we specify a source destination and a filename. In this case let's copy 'myfile' to the '/tmp' directory like we did before, but let's rename it as we do and name it 'copiedfile'. If we run the 'ls' command on the /tmp directory, we see the original file we copied before and a new version of that file with a different filename, copiedfile.

Move Files 4:21-7:17

There may be times when you want to not copy a file from one location to another in the file system but actually move it from one location to another in the file system. When you move a file, the source file is deleted and a new copy is created in the location that you specify.

You do this using the mv command; mv stands for move. In this example, we're going to move this file right here, mynewfile from my home directory over to the /tmp directory. We enter 'mv' and then 'mynewfile', the source file that we're going to move, and again we're using a relative path. Then we specify where we want to move it to.

We'll use an explicit path '/tmp', Enter. If we do an 'ls' of the /tmp directory, we see that 'mynewfile' exists over there. If we do an 'ls' of the local directory we see that mynewfile no longer exists where it did before in my home directory; it's been moved.

Just as with copy, you can also rename files as you move them. You do it in the same way you just explicitly name the file when you run the mv command. Let's enter 'touch mynewfile'. Create a new file with the same filename as before, 'mynewfile'.

Let's run the 'mv' command just like we did before, but this time we're not going to just specify a location but we're going to include a new filename as well: 'movedfile' and Enter. Let's do an 'ls' command of the /tmp directory. You see that movedfile is over there now. If we do an 'ls' of the local directory, we see that mynewfile is gone, of course, again because we moved it.

One of the neat things about the mv command is that you can use it to rename files without actually moving that file anywhere else. In other words, we're just going to use it to rename a file in place.

To do this you just specify a new filename that is in the same directory as the source file. The source file will be removed and a copy of it will be created with the new filename.

For example, we have a file here called myfile, let's suppose we want to rename it; we want to rename it renamedfile. So we enter 'mv', space, and then the name the file that we want to rename. Then we specify the new name that we want to assign to that file, such as 'renamedfile'.

Notice that I am using a relative path in both locations. We are going to take the mynewfile file in the home directory and we're going to move it to the same directory because we did not provide any explicit path information here. It's going to go into the same directory where it came from, but with a new filename. If we do an 'ls' command we see the file called renamedfile.

Delete Files 7:18-9:05

With that, let's talk about deleting files. You delete files from the shell prompt using the rm command. The rm command, by default, is used to delete files. It doesn't like to delete directories unless you specify -r, in which case it will delete directories.

Before we talk about deleting files with rm, it's very important that you understand that this is kind of a dangerous utility. You have to be careful. First of all, it's not going to prompt you by default before it deletes files. They will just be gone. Second of all, the file will not be placed in the recycle bin where you can get it back if you need to.

If you run rm and delete a file, it's gone. Let's use the rm command to remove this file right here, renamedfile. You simply type 'rm', and then the name of the file that you want to get rid of: 'renamedfile'. That file is now gone. Remember, I said that it will delete files without prompting you by default.

You can tell the rm utility that you do want to be prompted before deleting a file, and that's really a best practice especially if you're deleting a lot of files, because it's very possible that you may have inadvertently specified a particular file to be deleted in the list that you don't really want to be deleted.

Let's go ahead and create a new version of that file: 'touch renamedfile', so we have the file back now. This time let's use the 'rm' command with the '-i' option, i stands for interactive, meaning it's going to ask you before it deletes each file to confirm that you really do want to delete that file.

Let's delete 'renamedfile' again. This time notice that I'm prompted where I was not prompted before. "Do you want to delete this file?" I enter 'y' and now the file is gone. You can see it's been removed.

Get Information About Files 9:06-10:40

The last thing we're going to look at in this demonstration is using the file command. You can use the file command to view information about a particular file. Notice right here that we have a file called 'mynewfile2'. It does not have any kind of extension to tell us what kind of file that is.

We can even run a 'ls -l' command to view a little more information about it. Now we know that it's a 0 byte file, but this output does not tell us what kind of file this is. Let's make this file a text file, because right now it has 0 bytes. Therefore, it really isn't any kind of file in particular.

Let's use vi to add contents to 'mynewfile', save our changes and 'exit' the editor. At this point, mynewfile2 does have some content to it. Let's run 'ls -l' again. Now we see that the file size has increased. It's got 31 bytes, but we still don't know what kind of file this is. Is it a text file? Is it a spreadsheet file? Is an executable? We don't really know based on the output of the 'ls -l' command.

What you can do is run the 'file' command and specify which file you want to look at. Let's look at 'mynewfile' and when we do, the file command will analyze it and tell us what kind of file it is. As you can see here, mynewfile is an ASCII text file because we used vi to go in and add content to that file.

Summary 10:41-10:55

That's it for this demonstration. In this demo we talked about managing files from the shell prompt. We first talked about creating files. We then talked about copying files. We reviewed how to move files and to rename files. Then we talked about how to delete files. We ended this demonstration by talking about how to get information about files.
