2.1.5 Work with the Linux Shell

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

Work With the Linux Shell 0:00-0:18

In this demonstration, we're going to look at several key features of the Linux shell. Specifically, we're going to explore command history, we're going to use tab complete, and then we're going to look at several key path issues that you have to be aware of when you're using the Linux shell.

Command History 0:19-1:10

Command history is a really useful feature of the Linux shell. Basically, it allows you to reuse commands over, and over, and over without having to manually type them every single time you need to use them.

If we do an ls command (the ls command is used to list files) and if I don't specify a directory, it's just going to list the files in the current directory, which, as we saw earlier, is /home/rtracy.

I press Enter. And when I do, a list of all the files and folders in my user's /home directory is displayed.

By default, ls displays only non-hidden files. But within my /home directory there are actually a whole bunch of hidden files that we cannot see with ls by default. If I want to view regular files and directories as well as hidden files and directories, I enter the 'ls' command with the '-a' option for all, and it will display all the files, including the hidden ones.

Hidden Files 1:11-1:34

Notice here that there are many other files within my /home directory. All the hidden files by the way, just as a hint, begin with a period. If you see a Linux file or directory and it begins with a period, such as these files here and these directories over here, then you know that it is a hidden file or directory.

The directories over here are listed in blue; the regular files are listed in white.

Bash History 1:35-2:25

Notice that one of the files here is the .bash_history file. Every time I enter a command at the shell prompt, such as these, that command is automatically added to this file.

That way, if I need to run one of those commands again, all I have to do is press the Up and Down arrow keys. When I do, that bash history file is read and the various commands that I have typed in the past are displayed every time I hit the Up arrow key in the order that I originally entered them.

For example, if I wanted to run the uname -a command again, I could use the Up and Down arrow keys until I reach the uname -a command, press Enter and the output is displayed. I didn't have to type the command all the way out.

This can be really useful in situations where you need to type the same command over and over--especially if it's a really long command.

The history Command 2:26-3:04

If you want to see what commands are currently in your history file, you can use the history command.

Just type 'history' at the shell prompt and a list of all the commands that you have entered recently that are in the bash_history file are displayed, and these are listed in the order that they will appear if I press the Up arrow key. First go to history, uname -a, ls -a, ls, and so on.

If you have a lot of output currently displayed at the shell prompt and you want to clean things up and start over fresh with a clean screen, you can use the 'clear' command. It's very useful. That's how command history works.

Tab Complete Feature 3:05-4:22

The Linux shell also provides a very useful feature called tab complete and this is also very useful in situations where you have to type very long file names or very long directory names or very long commands.

Basically, how tab complete works is you start typing a command or filename at the shell prompt and instead of typing it all the way out, you just type the first few characters and then press the Tab key.

And when you do, the Linux shell is going to look in either the current directory or within the directory that you explicitly specified within the command and then try to find a matching file directory or executable.

For example, let's suppose that I need to change to the /etc directory in the file system. To do this, I would type the 'cd' command for change directory, space, then I can start typing the name of the directory that I want to change to. I can do '/e' and then if I press Tab, it automatically fills out the rest of the directory name for me: /etc/.

It can do that because there's only one directory at the root of the file system that begins with an e--that's the /etc directory. Then I can press Enter and I'm changed to the /etc directory. Let's switch back to tilde (~).

I'm going to press the Up arrow key to use my command history until I get back to the cd ~ command.

Multiple Matches 4:23-5:43

I'm back in my /home directory. There may be situations with tab complete where there are multiple matches. In this example right here there was only one match. There was only one file or directory at the root of the file system that began with e: /etc.

If there are multiple matches, then something different happens.

For example, I'm in my /home directory and I'm going to type 'cd Do'. I want to switch to my /downloads directory, so I type 'Do' and then I press Tab. Well, nothing happens. The reason nothing happens is because there are multiple directories or files that begin with Do.

If you want to see what they are, you can actually press the Tab key two times. If I press the Tab key twice, notice that it tells me that there are two matches. One is documents and one is downloads, so if I want to continue using tab complete without having to type the whole name of the folder out that I want to switch to, I can just add additional characters, which will narrow down the matching files or folders.

In this case, the first two characters of these two folders are the same, Do, but notice that the third character is different. If I type 'cd Dow' and then press the Tab key, there's only one match now and it is the downloads folder. Press Enter to switch to that folder. Let's switch back to my rtracy user's /home directory.

PATH Issues 5:44-7:19

The last thing we talk about in this demonstration are path issues.

When I type the 'ls' command to view a listing of all the files in my /home directory, notice that there is one file here that's in green instead of blue or white. That's because this is an executable file; it's actually a script that's been made executable. I can run this file from the shell prompt.

On most operating systems, such as Windows or even on the old DOS systems, to do this--because I'm in the same directory where this file exists--I would just type 'helloworld' and it should work, right? Press Enter here. It doesn't work on Linux. It fails. It says, "Hey, I can't find this file."

This is one of those things that drives new Linux users crazy because you look and you think, "Okay, the helloworld file that I want to run is in the current directory, yet when I run the command it tells me that it can't find the file. What is up with that?"

Well, in other operating systems, such as Windows, in the Windows command environment the command interpreter will, when you type a command, first look in the current directory for the command that you entered at the prompt.

If it finds it, it will run it. If it doesn't find it, then the Windows command interpreter will actually look for that command in all of the directories within the path environment variable, which makes this really easy.

The thing you have to remember is that the Linux shell does not do that. It does not look in the current directory when you execute a command at the shell prompt; it looks only in the directories within the path environment variable.

Explicit Directory Use 7:20-8:08

If your current directory, in this case /home/rtracy, is not found in the path environment variable, then it cannot run the executable. What do you do in that situation? Well, there are three different options.

One option is to explicitly specify the directory where the executable file resides, in which case then the Linux shell can find the file and run it. In this case, I can enter '/home/rtracy/helloworld' and if I run it, it works this time. That's because I told the Linux shell explicitly where that file resides, /home/rtracy.

Actually, to make things shorter I could have also just specified '~/helloworld' and it would still work, because remember the tilde is a shortcut that points to the /home directory of the currently logged in user.

Current Directory Use 8:09-8:46

Another option is to specify the current directory, whatever it happens to be. This can be useful if you're dealing with an executable that's in some folder somewhere in the file system that's buried many layers deep and you really wouldn't want to have to type out the full path.

You can use the ./ shortcut to specify that the command you're about to run exists in the current directory. ./ means the current directory, so I can enter './helloworld', and it runs because the Linux shell knows that ./ means current directory, which is /home/rtracy in this case.

The PATH Environment Variable 8:47-9:29

There's one other option that we could do, and that is move this file that we want to run to a directory somewhere in the file system that is in the PATH environment variable. If I were to come down here and use the echo command and then $ and then PATH, we can actually view the contents of the PATH environment variable.

The echo command simply displays text on the screen. If we enter 'echo' and then '$' and then 'PATH', it tells the echo command, "Hey, go out and grab the contents of the PATH environment variable and just write it on the screen." Let's go ahead and do that.

You can see there are several different directories that are included. Each directory is separated by a full colon (:), so this is the first directory in the path, second, third, fourth, fifth, and so on.

PATH Folder 9:30-10:46

But notice down here that there is a directory here within my /home directory --my user's /home directory --called bin, /home/rtracy/bin.

That directory is in the PATH environment variable, so what we can do is actually copy this helloworld file to the /home/rtracy/bin folder. Because it's in the path, we should be able to run it directly out of there without writing a full path to it.

If we do an ls command, we should see that the bin directory doesn't exist. When we do, we can see that a bin folder doesn't currently exist in my /home directory. So let's go ahead and make it; let's use the 'mkdir' command and create a folder called bin. If we do an 'ls' command, we have a folder called bin that matches the path to the folder specified in the environment variable.

Let's use the 'cp' command to copy 'helloworld' to the 'bin' folder. If we use the 'cd' command to switch to the 'bin' directory and do an 'ls' command, we see that the script file exists there. Now I can type 'helloworld' and it runs. I didn't have to specify the full path to the file, because the directory where that file exists is in the PATH environment variable.

Summary 10:47-10:56

That's it for this demonstration. In this demo, we talked about using key features of the Linux shell. We first looked at command history. We then looked at the tab complete feature, and then we ended this demonstration by talking about path issues.
