2.8.2 Navigate Directories

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

Navigate Directories 0:00-0:11

In this demonstration, we're going to look at several shell commands that you can use to navigate directories within the Linux file system.

pwd Command 0:12-1:18

The first one we need to look at is the pwd command. pwd stands for present working directory or (depending upon who you talk to) print working directory. It's a very basic utility. I press Enter. You can see that all it does is display the current working directory on the screen. pwd is of marginal usefulness because most Linux systems (most distributions) by default configure the shell prompt to display the current directory within the prompt itself.

You can see that that's the case with this system, here. This little tilde (~), right here, points to the home directory of whatever user is currently logged in to the system. So, many times, we can use that information to determine what our current working directory is. However, not all distributions are configured this way, although you can manually configure it if you wanted to. If it's not, then you can go ahead and use pwd to quickly see what the current directory is. Also, be aware that the directory that's usually displayed in the shell prompt is only the current directory. It doesn't display the full path to that directory. And, therefore, pwd can be useful to see what the full path to your current directory is.

ls Command 1:19-2:51

Okay. So, the next command we need to look at is the ls command. This is an extremely useful command that I promise you will use all of the time. The ls command displays the contents of a specific directory. It displays all the files and all the subdirectories within that directory. Currently, I am in my home directory, /home/rtracy. If I just enter the ls command, it'll display all of the files and all of the subdirectories within my home directory. As you can see, the output is color-coded. This will not be the case with all distributions, but many distributions do use color-coding by default within the shell environment, so you can quickly identify what type of entry each one of these is.

The colors used may vary, but generally speaking, any item displayed by ls in blue is a directory. As you can see, I have many subdirectories within my home directory--Documents, Downloads, Music, Pictures, and so on--and any standard files within that directory are displayed with either white or black. My shell environment is configured with a dark gray background; hence, regular files are displayed in white. But notice, over here, that we do have a standard file, but it's colored green. That's because it's a special type of file.

This file here, myfile, is just a standard file. It could be a text file that you put characters in. This file over here, however, is colored green because it's an executable. It's actually a script file that can be run from the shell prompt. And usually, executable files are identified in green.

Before we go any further, we need

Relative and Explicit Paths 2:52-3:17

to discuss the difference between relative paths and explicit paths. And this concept is going to be used with pretty much all of the commands that we're going to look at in the rest of this demonstration. Notice, up here ,I just typed 'ls' with no directory specified. Therefore, the ls command assumed that I wanted to display a listing of files and subdirectories of the current directory, whatever that may be. However, you can explicitly

Explicit Paths 3:18-4:05

specify a particular path to use with the ls command. For example, if I wanted to view the contents of the /etc directory, I could enter 'ls', 'Space', and then an explicit path to the directory that I want to generate the listing of. In this case, it'll be /etc. When I do, lots of files and subdirectories are displayed. Notice that, even though I'm generating a listing of the /etc directory, my current directory is still the same. It's my home directory. I didn't change into that directory; I just went out and got the contents of that directory and displayed it on the screen.

Notice, when we typed that command, we used what is called an explicit path. We started at the root of the file system, /, and then specified the directory that we wanted to look at, /etc. You can also use relative directories.

Relative Paths 4:06-4:57

Let's run the ls command again from our home directory. Notice that there is a folder here called temp. If I wanted to view the contents of temp, I could type 'ls temp'.

If I wanted to use an explicit path, I would type '/home/rtracy/tmp', and that would work. That's just fine. But it requires a lot of typing. If we simply type 'temp', we're using a relative path. In other words, the ls command is going to say, "Oh, the user wants to view the contents of a directory named temp. Oh look, guess what? There's a directory named temp in the current directory. I'm going to assume that that's what the end user wants to do." If I hit Enter, it will display a listing of all the files in the /home/rtracy/tmp directory. This is called a relative path, meaning that the full path to this directory that we entered is relative to the current directory.

ls -l 4:58-5:57

Notice that, when we run the ls command, it only displays the names of the files. We don't see any other information about those files. In many situations, you're going to want to know additional information about these files and subdirectories. You might want to know how big they are, what the permissions are that are assigned to it, who owns it, and so on, in which case, you can use the ls command with the -l option. -l stands for long--it uses a long output format.

Press Enter, and you see that all of the same files and directories that were displayed earlier are displayed again. But this time, we see lots of other information about it. For each file and subdirectory, we see, first of all, the mode of the file, right here. The mode represents the permissions that have been assigned to that file that control access to it. We can also see the name of the user that owns that file. We can also see the name of the group that owns that file. We can also see the size of the file. And we can also see when the file was last modified. That's useful information.

ls -a 5:58-6:53

Understand that what we're seeing here are regular standard files and subdirectories. There are other files and subdirectories within the Linux file system that are not displayed by default by ls, especially in a user's home directory. There are many hidden files within this directory that are used to configure the user's computing environment, but they're not displayed by default by ls. If you want to see regular files as well as hidden files, you need to use the -a option with the ls command. -a stands for all.

Notice, now ,that many other files and subdirectories are displayed when you run the ls -a command. We can see that there are several hidden files, and we know that they are hidden because they start with a period. Any file or directory in the Linux file system that starts with a period is a hidden file or hidden directory. And here, you can see we have several hidden files, and we also have several hidden directories as well.

Combine Options with ls 6:54-7:19

You can combine other options with the ls command. For example, let's suppose we want to view extended information about all regular and hidden files within the home directory of my user account. We can use the -a option with the -l option. And when we do, we see extended information about both hidden and regular files in my home directory.

That's how the ls command works. Let's clear things out with the clear command.

cd 7:20-9:51

The next command we want to look at is the cd command. cd stands for change directory. It's basically used to switch between directories in the file system. If we do an ls command real quickly, we see that we do have a subdirectory within my home directory called temp. If I wanted to change into that directory to make it my current directory, I would type 'cd temp'--change directory to temp--and when I do, notice that the prompt, over here, changes because now my current directory is temp. After I run the pwd command, we see that my current directory is /home/rtracy/tmp now.

Notice, here, again, we use a relative path. Because we didn't explicitly specify where temp is, the cd command assumed that we were talking about the /tmp directory within the current directory. I could have used a full explicit path if I wanted to, so that's cd /home/rtracy/tmp, and that would've worked as well; it just would've required a lot more typing. There are situations when you do need an explicit path. For example, let's suppose we wanted to change into the /tmp directory. In order to do this, I have to specify the full path to that directory. / is the root of the file system, and the name of the subdirectory of the root file system is tmp. Hit Enter, and I'm changed over to the /tmp directory at the root of the file system.

This is an explicit path because we start at the root of the file system, and we work our way down to the particular directory that we want to change to. Let's go ahead and switch back to my home directory. I could use, again, an explicit path to change to my home directory, /home/rtracy. Or I could just specify ~. Remember, ~ is a shortcut specifying the home directory of whatever user is currently logged in. I do 'pwd' again. I'm back to /home/rtracy.

There are other shortcuts you can use with the cd command in addition to ~. For example, let's suppose I wanted to go up one level in the file system. I want to go to /home. One option would be to type 'cd /home', and that's fine. That will work. It's more typing though. The easier way is to type 'cd ..'. using .. tells the cd command that we want to go up just one level in the file system. If I hit Enter, my current directory is now changed to /home. .. is, basically, a shortcut to the parent directory of whatever the current directory is. Let's go back to my home directory here.

Summary 9:52-9:55

That's it for this demonstration. In this demonstration, we looked at several different commands that you can use to navigate the Linux file system.