2.4.2 Use Aliases

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

Use Aliases 0:00-0:17

In this demonstration, we're going to talk about using aliases. The alias command is used to manage aliases within your shell environment. In fact, several different aliases are automatically defined for you whenever you log in to your Linux system.

View Aliases 0:18-0:42

You can view them by typing 'alias' at the shell prompt. When we do, we see that I have many aliases that were defined for me when I logged in to the system. For example, there is the l. alias, which runs the ls -d . command. We also have the ll alias, which simply runs ls with the -l option, which displays the output in long format, and so on.

Define New Aliases 0:43-1:14

You're not stuck with these aliases that were automatically defined for you. You can add and remove aliases as needed. For example, let's suppose that we frequently need to run the ls -al command. As you know, the ls command is used to display files and directories in the file system. The -a option with the ls command causes ls to display regular files, as well as hidden files. The l option causes the ls command to format the output of the command in long format with extended information about each file and directory.

Name=Value Argument 1:15-2:37

To make things a little bit easier, let's suppose we want to define an alias where all we have to do is type 'lll' and it will automatically run the ls -al command for us. To do this, we can type 'alias', and then we enter the name of the alias we want to define--in this case 'lll'--and then we enter an equals sign (=) to define the command that that alias will actually run. We enter quotation marks (""), and then, between the quotation marks, we enter the actual command, 'ls -al'. Press Enter. The new alias is defined. We can verify that it's there by typing 'alias' again, and we should see that we now have an alias called lll that runs ls -al.

Let's go ahead and see if it works. Let's type 'lll'. When I do, the files and directories that reside in the current directory, which is the home directory for my rtracy user. I see regular files as well as regular directories, but I also see hidden files and hidden directories, which is exactly what I wanted. The output of the ls command is formatted in long format, where I can see--for example, here, with this file--the mode of the file, the user that owns the file, the group that owns the file, and the modification date of the file, as well as its size, right here.

Persistent Aliases 2:38-4:09

It's important to note that even though I've defined this alias, it is not persistent. If I were to reboot the system, the lll alias that I just defined would be gone. In order to make it persistent, we need to add that alias command that we just typed to the appropriate shell configuration file. The file that you will use depends upon which distribution you're using. Notice, for this distribution, which is Fedora, that we have a .bashrc hidden file in my home directory. This is the file I would modify to automatically create that lll alias every time I boot the system.

Let's go ahead and do that. I'm going to use the vi editor to edit the .bashrc file. Press the Insert key to go into insert mode. And if I arrow down, notice that the last line in the file says, "User specific aliases and functions." This is where you can define you own custom aliases and functions, so we enter the command to do just that. Let's go ahead and enter the command that we entered before, 'alias lll="ls -al" '. Press Escape and enter ':'. enter 'exit' to exit the vi editor and save my changes to the file. Every time I boot the system, the lll alias will be automatically created for me.

Notice in the lll alias that we just defined that only one single command is run when we type lll at the shell prompt.

Aliases with Multiple Commands 4:10-5:16

You're not limited to just one command with an alias. You can actually associate an alias with a series of commands to perform some task. For example, if we look in my home directory, notice that there is a folder called temp. Go into temp and do an ls command. We see that I

have nine or ten different temporary files that I create throughout the day as I go through my work. They're just temporary files. They're nothing really important, so I need to clean these out now and then.

If I were working and wanted to clean those folders out, I would have to, first of all, type the 'cd temp' command. I would use the 'rm *' command to delete all the files. And then I would type 'cd ~' to go back to my home directory. In order to clean up the temp directory, I would have to type three different commands. We can speed things up and make things more efficient by creating a single alias that will do all three of these commands all at once. To do this, we enter 'alias' again, just like we did before, and then we need to define the name of the alias, just like we did before. This time, instead of lll, let's name it cleanup.

Use Quotes(") and Semicolons (;) 5:17-6:49

You do an equals sign (=). Then we enter a list of commands enclosed in quotes ("") that we want this alias to run for us. The first thing we need to do is to change to the ~/temp folder. Remember, the tilde specifies my user's home directory. We want to go to the /temp directory inside of my home directory; then we need to run the rm command to clean things out. We need to separate that second command from the first one in the alias using a semicolon (;). I enter ';' and then I use the 'rm -v', for verbose so we can see what's actually happening when the rm command runs. And then we have to specify which files we want to delete within the temp directory.

Let's just use an asterisk (*), a star, as a wildcard to specify that all the files in the directory be removed. That's the end of the second command, so I enter another semicolon (;). At that point, I probably want to switch back to my home directory out of the temp directory. I'm going to type 'cd ~'. These three commands are going to be run, cd into temp, delete all the files, and then cd back into my home directory. Hit Enter. Type 'alias' again to view a list of all my currently defined aliases. And there's my new alias that we just defined right there, 'cleanup'.

Let's test it out. Let's type 'cleanup' at the shell prompt, and you can see that the command ran and, as each file was deleted, a message was displayed on the screen. We go into the temp directory. Do an ls command. You can see that all the files are gone.

Remove Aliases 6:50-7:33

There may be times when you have an alias currently defined on your system, but you don't actually want it. You may want to get rid of it. For example, let's say we decide that we really don't want the cleanup alias anymore. We've moved on to a new project. We're not using the temp directory anymore, so let's just get rid of the cleanup alias that we've defined.

To do this, you enter 'unalias' at the shell prompt followed by the name of the alias you want to delete, 'cleanup'. I run 'alias' again, and you'll note here that the cleanup alias is no longer displayed. If I try to type 'cleanup' now, nothing happens. The shell doesn't know what to do.

Summary 7:34-7:50

That's it for this demonstration. In this demo, we talked about managing aliases at the shell prompt. We talked about how to view a list of existing aliases. We talked about how to define a new alias that contains one command. We talked about how to define an alias that contains multiple commands. And then we ended this demonstration by discussing how to remove an alias.
