2.7.6 Command Substitution

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

Command Substitution 0:00-0:23

The BASH shell allows you to use command substitution, which is really useful because it means you can run a command and have its output pasted back onto the command line as an argument for another command. Essentially, command substitution allows you to perform multiple tasks at once with a single command.

How Command Substitution Works 0:24-0:54

Command substitution works by first creating a child process that runs the first command. The standard out from the command is then piped back to the BASH shell. The shell parses the output from the first command into words that are separated by white space. After the pipe from the first command closes, because the command is done running, the shell then starts another child process to run the second command using the standard out from the first command as arguments. Let's take a look at an example of how this works.

Case Study: tail Command 0:55-1:55

In this example, we need to use the tail command to view the last few lines of all files in the /etc directory that contain the text ens192. ens192 is the name of our network interface. Essentially, what we want to do is search all the files in /etc. If a file contains the text ens192, the name of our network interface, then we want to grab that file name. Then we want to send a list of all those file names that have that matching text as input to the tail command so we can view the last few lines of that file. If a file does not contain ens192, then we're not concerned about looking at it.

You can do this in two different ways. The first way would be to manually run first the fgrep command, and then the tail command. You'd have to pipe the output from the fgrep command to a file and then send the name of the file's input to the tail command, and that's one way it would work. It would just require a little more work.

Solution: Command Substitution 1:56-3:22

Another way is to use command substitution. An example of how to do that is shown here. First we enter 'tail'. Then we enter a dollar sign ($). And then, in parenthesis, we enter the first command that we actually want to run. In this instance, the first command to be run is the fgrep command, and we're going to tell it to search through all the files in the /etc directory, and the text string that we're looking for is ens192.

Notice, over here, that we use the '-l -r' option with the fgrep command. This does two things. The -l option causes fgrep to just return a list of file names that just contain the matching term, not the actually matching text itself. By default, fgrep will return the matching text. We don't want that in this case. We just want the file name that has the matching text. The -r option tells it to look through all the subdirectories of the directory we specify over here, which is /etc.

Essentially, by doing this, we generate a list of files that contain the matching text. This list of files is then sent as input to the tail command. We're basically giving tail a whole list of files that it needs to go and look at, and it will then display the last few lines of each file in the list, the files that it receives from the fgrep command.

Command Substitution vs the xargs Command 3:23-4:23

In addition to command substitution, you can also use the xargs command within the BASH shell to accomplish a similar thing. Here's the issue that you're going to run into: it is possible, depending upon how it's used, that command substitution could fail. It will do this if that first command that we looked at pipes too many results back to the second command. In this situation, you end up with an 'arguments list too long' error. This happens because of a limit that's imposed by the Linux kernel.

Here's the key thing you need to remember: the maximum length of a BASH command is 128 kilobytes. Any command that's longer than this causes this error to be displayed. In this situation, you can use the xargs command instead of command substitution. This works because the xargs command will actually break down a long command line into 128-kilobite chunks and then pass each chunk one at a time, as an argument, to the command specified in the xargs command line.

Case Study: find and rm Commands 4:24-5:24

For example, let's suppose that you use your Linux system as a productivity system. You have an office suite installed on it, and you create Word processing documents, presentation files, spreadsheets, and so on, and you use your system a lot. As a result, your home directory is getting choked up with all the backup files that are being created by your office applications. Now, it's important to know that on Linux, these backup files that are created by these office applications usually end in a tilde (~).

You need to clean things up. You've got tons of files in your home directory that end in a tilde (~), and you don't need them. In order to do this, to clean things up, you want to first use the find command to search through your home directory and create a list on all the files that end in a tilde (~).

Then you want to use the rm command to delete them. Could you use command substitution to do this? Yes, it could work, unless you have so many files in your home directory that you exceed that 128-kilobit limit, which is a possibility.

Solution: xargs Command 5:25-6:24

To prevent that from happening, we can accomplish the same thing using the xargs command. Here's how we would do it. First, we run the find command, and we tell it to start looking in the home directory. What we're going to do is try to find any files within the home directory, including all the subdirectories of home, that end in a tilde. The output of the find command will be a list of matching files, any file that ends in a tilde.

We're going to pipe that output from the find command to the input of the xargs command, which will then process that text stream 128 kilobits at a time and send it to the standard input of the rm command, basically giving it a list of files to delete one at a time. Even if there are tons and tons of files that would exceed the 128-kilobit limit imposed by the shell, we're still all right because xargs is going to divide up the input into those chunks that won't exceed that 128-kilobit limitation.

Summary 6:25-6:31

That's it for this lesson. In this lesson, we talked about how to run multiple commands at the same time with a single command at the shell prompt using command substitution, and also using the xargs command.