2.7.2 Piping

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

Piping 0:00-0:38

In this lesson, we're going to talk about using piping with Linux commands. Piping allows you to take the output from one command and, instead of writing it on the screen, we redirect it to the input of another command for it to process. Pipes can be used with any command that produces output of some sort and can accept input of some sort. You implement piping using the 'l' character. Basically, the pipe character tells the Linux shell to redirect the output from one command away from the screen into the input of some other command.

Piping the cat Command 0:39-2:07

Let's take a look at an example.

Let's say we want to use the cat command, here, to view the contents of our boot.log file, which is located in the /var/log directory. You can do this by simply typing cat /var/log/boot.log. But there's a problem with this command. The issue here is that boot.log on most systems is a very, very large file. When you view it with the cat command, the cat command simply displays all the text at once on the screen, and it's going to be far more text than will fit on the screen all at once. If we just use cat to view this very large log file, then the output is going to run off.

To fix this, what we can do is take the output from the cat command and, instead of displaying it on the screen, we use the pipe character to pipe the output from cat to the input of another command that does pause the content one page at a time. In this case, we're going to use the more command. We would enter cat/var/log/boot.log to display the text in the file, and then we're going to redirect the output from the screen to the input of the more command. The more command will then display the text on the screen, but it will pause it one page at a time, which gives us the opportunity to scroll through it one page at a time until we find the information that we're looking for.

Piping the journalctl Command 2:08-2:55

Now, piping is extremely useful, especially when you're troubleshooting issues. For example, let's assume that the Linux system, here, is having some network connectivity issues. We need to look in our system log file to see if we can figure out what's going on. Specifically, we're going to look for all entries in that log file that contain the name of our system's network interface, which is ens192. If you're crazy, you could just run the journalctl command, right here, by itself, and it will display the contents of the entire log file. journalctl is really nice because it does pause the output one page at a time, unlike the cat command. I could then manually review the file line by line, screen by screen, until I find the information that I needed. Thanks, but no thanks. That would take forever.

Piping Using the grep command 2:56-3:55

A better option would be to go ahead and run the journalctl command, like we were talking about before. But instead of writing the output on the screen, we could instead pipe the output to the input of another command called grep. The grep command will then take that input, and then it will search through that input for the specific text that we specify--in this case, ens192. Any entries in that log file that match will then be written to the screen. Any entries in the log file that do not contain ens192 will be skipped.

You can see the sample of the output right here. You can see here that the only lines that are displayed from the log file are those that contain the text ens192. Instead of having to search through miles and miles and miles of log entries, I can search through just a limited number of log entries to see if I can figure out what's causing my network problem.

Multiple Pipes in One Command 3:56-4:51

One of the cool things about piping is the fact that we can include multiple commands within the same pipe. Basically, this allows you to take the output of one command and send it to the input of another command, just like we did before. But then we take the output of the second command and pipe it to the input of yet another command.

In this example, we're going to use the journalctl command to display all the contents of our system log, and we're going to pipe it to the grep command, looking for the search term ens192, just like we did before. The output is probably still going to be quite long, especially if the system's been in use for a while. We could, therefore, pipe the output from the grep command to the input of the more command so that we

pause the output a page at a time, which gives us time we need to look for the specific information we need. We enter —˜journalctl | grep ens92 | more'.

Piping to the tee Command 4:52-5:53

Occasionally, there may be times when you want to take the output from one command, display it on the screen, and write that information to a file in the file system at the same time. You do this using a command called tee.

The syntax for using tee is to first run whatever command it is you want to run and then pipe the output from that command to the input of the tee command. Then specify what file name you want to write the output from tee to. In this example, I'm going to run the ls -l command, and I want the output from ls -l to be written on the screen, as you see right here, and I also want it to be saved to a file named files.txt.

Using the tee command, we can do this. We take the output of the ls command and send it to tee. The output is written on the screen, and then it's also written to a file, in this case, files.txt.

Summary 5:54-6:10

That's how piping works. With piping, we take the output from one command, and we pipe it to the input of another command. We can even include multiple commands within the same pipe to form what's called a text stream. And we also talked about how we can also use the tee command with piping to take the output from one command and split it, sending it to a file and to the screen at the same time.
