11.1.1 Linux Log Files

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

Linux Log Files 0:00-0:21

Let's spend some time learning about Linux system log files. Now, you need to understand that log files are a gold mine of information for the system administrator. For example, you can use your log files to detect intruders who are trying to break into your system. You can also use log files to troubleshoot problems with your system.

Log File Storage Location 0:22-0:55

Your system log files are stored in the /var/log directory. Notice there are many files and many sub-directories within /var/log where your system daemons are going to store their log files. Some of these log files are plain old text files that you can read with any text manipulation utility, like cat or less. Some of these files, on the other hand, are binary files. And the only way you can look at them is to use a special utility designed specifically to read that log file.

Common Log Files 0:56-3:12

With most Linux distributions, there are many log files stored within /var/log. As with anything else, some of these log files are really, really useful, and others, not so much. Some of the more important log files are listed here. The first two are basically the same, boot.log and boot.msg. Some Linux distributions use boot.log, and others use boot.msg. Both of these contain log entries that were created as the system booted up. Depending upon which distribution you're using, either one of these files can be a very valuable troubleshooting tool when you're trying to figure out start-up problems. We also have the cron log file that contains messages that were created by the cron daemon. We also have the dmesg log file, which contains hardware detection information. We have the faillog file, which contains failed authentication attempts--very useful if you're trying to detect intruders.

We have firewalld. This contains log messages created by the system firewall. We have lastlog, which contains information about the last time each user logged into the system. Depending upon your distribution, you'll see either mail or maillog. Both of these files contain entries that are generated by the mail daemon, the MTA on your system. We also have the messages file. Now, messages are used only on older Linux distributions that run the init daemon. The messages log file on these distributions is the main log file; this is where a lot of very useful logging information is stored. It's a simple text file and can be viewed with any text manipulation utility. Newer distributions use the systemd daemon instead of init, and it stores the main log messages in the journal instead of in the messages file. We also have secure, which contains information about access to network daemons.

We also have the rpmpkgs log file, which contains a list of installed RPM packages. We have the warn file, which contains warning messages. We have wtmp, which contains a list of users who have authenticated to the system. And we have xinetd.log, which contains log entries from the xinetd super daemon.

View the System Log Files 3:13-4:28

So, with this in mind, let's discuss how you go about viewing your log files and using them to troubleshoot problems.

As we said earlier, your log file can be an invaluable resource when you're troubleshooting Linux problems. If the kernel or some daemon running on the system encounters a problem, most likely, that problem's going to be logged to a log file. Reviewing your log files can provide you with a wealth of information that probably won't be displayed on the screen. As I said just a minute ago, on a systemd-based distribution, your system log messages, which comprise most of the log messages that are going to be generated on your system, are going to be stored in the journal. You view the journal using the journalctl command.

If you need to troubleshoot a problem associated with one specific application or service running on the system, then you might need to go beyond the journal and look at the log file that's maintained specifically for that problematic service.

For example, you would look in the maillog file on a Fedora system to troubleshoot problems with the postfix daemon. Or, if you were having trouble with the mysqld daemon running on your system, you would look in the mysqld.log file, which is located in /var/log/mysql.

Search System Log Files 4:29-6:33

There's one key problem you need to be familiar with when you're looking at log files, and that is the fact that log files tend to become very long. It's very difficult to find specific information just by browsing through them.

There are two strategies you can use to get around this problem. The first one is shown here, where we redirect the output of the command that's used to view the log file to the grep command, and then we use the grep command to filter out just one specific term within the log file. In this example, we needed to locate information in the /var/log/boot.log related to the starting of the OpenSSH daemon when the system booted up. To do this, we run 'cat /var/log/boot.log' and then we and pipe the output to grep and tell it to look for the text 'OpenSSH'. When we do, only the lines within that log file that contain the term OpenSSH are displayed in the output of the command. Basically, this takes a log file that contains hundreds of lines in it and condenses it down to these two, and these are the two that I needed to look at.

If your system uses systemd in the journal, then you can do the same thing when you're searching through the system log file with the journalctl command.

All you have to do is run 'journalctl -e' and pipe the output to the grep command and tell it what you're looking for. The -e option is important because it tells journal control not to pause the output page by page. By default, journalctl does; that's not what we want it to do. Instead, we want journalctl to start at the beginning of the journal, scroll all the way through it--sending that output to the input of grep--and tell grep, "Hey, I just want to find the lines that contain the text,— (in this case,'ens192'.

Only the lines in the journal that contain the text 'ens192' are displayed in the output of the command. That helps me narrow down the thousands of lines in the journal to just these few that contain the information that I need.

Monitor Log Files 6:34-8:41

You can also use the head and tail utilities to view log file entries. Understand that most log files record entries chronologically, starting at the oldest and then going to the newest. If you want to view the beginning of the log file, you can run the head command followed by the name of the log file at the shell prompt, and it will display the first few lines of the file.

Frankly, this is not all that helpful because the first lines in a log file are the very oldest lines in a log file. They're usually not all that helpful. Tail, on the other hand, is extremely useful because it works in a manner opposite of head.

Instead of displaying the first lines in a file, it displays the last lines in the file. This is useful because when you're troubleshooting, you usually only want to see the last few lines of the log file. We don't care what happened last Tuesday. We want to know what happened just a minute ago. One way you can do this is to run tail followed by the name of the log file that you want to view, and the last few lines of the log file will be displayed. There's an option that you can use with tail that increases tail's utility, which is (at least, in my opinion), a great deal. That's the -f option that you see here. I use this all the time when I'm troubleshooting problems on a Linux system.

When you use the -f option, tail will initially display the last lines of the log file specified as per normal, nothing new there. But it doesn't exit. What it does is then continually monitor that log file, and then it will display each new line as it's added to the log file. In this example, I've used the tail command to view the /var/log/firewalld log file--the log file for the firewall running on my system--and I've used the -f parameter. So, tail is going to continuously monitor the firewalld log file. And as new messages are added to that file, as I troubleshoot problems, they're going to appear on the screen.

This is good because the results are instantly displayed without requiring you to run tail over and over again. Once you're done monitoring the file, you can break out by pressing Ctrl + C.

Monitor the System Journal 8:42-9:18

If your system is based on the systemd daemon and uses the journald daemon to manage logging, you can do the same thing with the journalctl command while you're viewing the system log.

We run 'journalctl' with the -f parameter. This runs in exactly the same manner as if we were viewing the journal with the tail command, which we can't do because the journal is a binary file that can only be viewed with a journalctl command. The first thing it does is display the last few entries in the journal. Then it sits and monitors the journal and will add each new entry to the output as it's added to the journal file.

Summary 9:19-9:28

That's it for this lesson.

In this lesson, we reviewed some of the log files that are used on a Linux system. We first reviewed the names of your log files and where they're saved, and then we talked about using these log files to troubleshoot system problems.
