4.4.1 System Services

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

System Services 0:00-2:02

In this lesson, we're going to talk about managing system services. Managing boot targets is an important aspect of administering a Linux System. In addition to switching boot targets and configuring the default boot target, you also need to know how to specify which system processes and services will be automatically started at a particular boot target. In this lesson, we're going to talk about how to do this very thing.

Before we do so, though, we need to quickly review what a service actually is. You're probably already familiar with the concept of an end-user application that runs on an operating system. End-user applications are designed to interact with the end user and to help the user complete some type of productivity task. For example, if you want to run a word processing application like LibreOffice, then you're using an end-user application.

Now, because these applications have to interact with the end user, they usually provide some type of user interface, either a graphical user interface or a command line user interface. But system services (which, on Linux, are often called daemons), they're different. Like an application, a daemon also runs on the operating system, but they are designed to provide services that keep the system up and running and make it useful in some way. They're not designed to continually interact with the end user like an application does.

As such, they usually do not provide any kind of user interface. Some examples of commonly used Linux services include FirewallD that provides a firewall on the system; SSHD that allows remote access to the system; and NTPD, which allows the system to synchronize it's time with other computers on the network. There are many others. We won't go into them all here; there's hundreds of them. Just understand that if you need to start, stop, restart, or view the status of a particular service on the system, then you need to use the systemctl command.

Start and Stop a System Daemon 2:03-3:03

The syntax is to enter 'systemctl' followed by the action that you want performed, and then the name of the service that you want to perform that action on. For example, if you need to start a system service or a system daemon, then you enter 'systemctl start' and then the name of the service.

On some distributions, you have to put the full name to the service file for the service, which would be the name of service .service. On other distributions, you can actually leave this part off; you don't have to put the extension on. You'll have to try it on yours and see which one works.

For example, if we needed to start the firewall service, the firewall daemon on the system, we would enter 'systemctl start firewalld.service'. On the other hand, if you need to stop a service that's already running on the system, you use the stop action with the system control demand. You enter 'systemctl stop' and then the name of the service that you want to take down.

For example, if we wanted to stop the firewall service, which you probably should not do unless it's an emergency, you enter 'systemctl stop firewalld.service'.

Restart a System Daemon 3:04-4:55

There may be times when you need to stop a service and then start it again. One way to do this would be to simply enter the stop command followed by the start command, and that's fine.

If you want to do it with one single command, you can use the restart action with the systemctl command. You enter 'systemctl restart' and then the name of the service; that will take it down and bring it back up. This is very useful in situations where you're having problems with the daemon. Maybe you can't connect, for some reason, to a service, or whatever. You can take it down and start it again to see if it works after restarting. It's also useful in situations where you have made changes to the daemon's configuration file. If you do this, the daemon will have no idea that you made those changes because it only reads its configuration file when it initially starts up. So, one way to apply the changes is to use the restart command to shut the daemon off and bring it back up.

I do need to provide you with a word of warning in this respect. If the service that you are going to restart is one that is heavily used or currently in use--say the system is running as a web server, and you're to restart the apache2 daemon, which provides the web service--be aware that the service does go down if you use the restart action with the systemctl command. It's only down for about a second, maybe two at the most, but it does go down. This means that any connections that were established with that service will be cut off, and then they'll have

to be reestablished when the service comes back up. Be very careful when using the restart option. You may want to use the restart option late at night or some other down time when that service isn't being used.

An example is shown here. If we made changes say to our firewall configuration and we want those changes to be applied, we would run 'systemctl restart firewalld.service'.

Check Daemon Status 4:56-5:54

In addition to starting, stopping, and restarting a service, there'll be many times when you need to check the status of the daemon. You may want to know, is it running? Is it not running? Is it even installed on the system? In order to check the status of a daemon, you enter 'systemctl status' and then the name of the service that you want to check. For example, if you want to see what the status of the firewall daemon was on our system, we would enter 'systemctl status firewalld.service'. This is a very useful command. I use it all the time. As I said, it tells you if the daemon is running or not, and it also shows you the last few log messages that were generated by the daemon.

If you're having problems—"say, your web server isn't running properly--and you run the systemctl status command for the apache2 service, in the output of the command, you may see, in the log entries displayed, whether or not the system is having problems. It's not a cure-all for everything. You'll probably still need to look at your system log in order to determine what the real problem is. But sometimes it gives you a quick clue as to what the problem might be.

Reload Daemons 5:55-7:08

Remember, earlier, I said that one of the problems with using the restart command is that it takes that service offline for just a few seconds. And if you had connections already established with the service, they're broken, and they're going to have to be reestablished. Depending on what the service is, that could make end-users a mite upset.

Therefore, another option that is provided by some daemons is to reload instead of restart. The syntax is shown here. You enter 'systemctl reload' followed by name of the service. What this does is leave the daemon running, but will go and re-read that daemon's configuration file and make the appropriate changes, leaving that service up and running the whole time.

An example is shown here. We'd enter 'systemctl reload' and then the name of the daemon, in this case 'firewalld'. The benefit to this is that any established connections with the service involved would not be broken. The sad thing is, not very many daemons actually support the reload option. A lot do, but many, many don't. You can run this command with a given daemon and just see if it works or not. If it does, you know that the daemon supports it. If you get an error, you know that that daemon does not support the reload action with system control.

Enable Daemons 7:09-7:43

You can also use the systemctl command to either enable or disable a particular daemon on system boot. If it's enabled, then when the system boots up, it will automatically be loaded. If it's disabled, then the daemon will not load until you manually load it. The syntax is to enter 'systemctl enable' followed by the service name.

In this example, we have the firewalld service. It's a very important service. We want to make sure that it's always running by default on our system. We enter 'systemctl enable firewalld.service'. That way, every time the system boots, the firewall daemon will automatically run.

Disable Daemons 7:44-8:36

There are other daemons that you want to start manually on the system. You don't necessarily want them to start automatically when the system boots; you only want them to run when you tell them to do so. There may be some daemons on your system that you just don't want period, in which case you should actually uninstall them. That increases the overall security of the system.

Other daemons you will want there, but only want them to run when you decide that you need them. In this case, you can disable the service at system start. You enter 'systemctl disable' and then the name of the service that you want to disable. In this example, I'm doing something very stupid, and I'm disabling my firewalld daemon on the system. After running this command, the next time I boot the system, the firewalld service will not run until I manually do it. And it's probably a poor example. Don't ever enter this command on your Linux system unless you really, really want that firewall disabled.

Verify that a Daemon is Enabled 8:37-8:55

You may have a daemon on your system, and you're not sure whether it's enabled or not at system boot. To see if it's enabled, you enter 'systemctl is-enabled' followed by the name of the service. In this example, we're going to enter 'systemctl is-enabled firewalld.service', and it will tell us whether or not that daemon is going to start automatically when the system boots.

Block a Daemon from Starting 8:56-10:01

Here is an interesting aspect of the 'systemctl' command. There may be times when you have a particular daemon on your system and you want to leave it installed, but you do not want it to be booted at all, even manually. You don't want to start it automatically at system boot, nor do you want anyone to be able to start the daemon from the shell prompt. In this case, you can use the mask action with the system control command.

The syntax is to enter 'systemctl mask' and then the name of the service. This effectively blocks that daemon from starting, and you will not be able to start the daemon at all on the system until you unblock it. In this example, we're doing something, again, very, very foolish, and we are blocking the firewalld service. We enter 'systemctl mask firewalld.service', in which case the firewalld service will not start at system boot, and it cannot be started manually from the shell prompt.

When you come to your senses and realize that you really do need a firewall on your system, before you can load it, you would have to enter this command here: 'systemctl unmask firewalld.service'.

Summary 10:02-10:13

That's it for this lesson. In this lesson, we talked about how you manage system services using the 'systemctl' command. We talked about how to start, stop, and restart a service. We also discussed how to make sure a service starts automatically whenever the system boots.
