10.3.1 at Task Scheduling

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

at Task Scheduling 0:00-0:48

In this lesson, we're going to look at using the at daemon, or at. Using at is a great way to schedule a task to run once sometime in the future, say, a backup job.

The at daemon, which we call atd, runs in the background on your Linux system, checks its schedule, and waits for a particular task that needs to run. And when the time comes, it runs it for you.

Most Linux distributions will install this daemon for you during the initial installation of the system, but some don't. If this is the case, you may need to go and install it manually from your installation repository, using yum, zypper, or apt-get.

Before attempting to use at, you do need to make sure that the atd daemon is actually running on your system.

Start atd 0:49-2:08

If your Linux distribution is based on the init daemon, then the startup script used to start the atd daemon is located in your init script directory. And, depending upon your distribution, it could be /etc/init.d or /etc/rc.d/init.d.

Regardless of which directory your distribution saves your init scripts in, the name of this script is just atd. And you can run it right out of your init script directory, or you can also run it using the commands that you see here, rcatd, and then you specify start to start the daemon up.

It's important to make sure that the atd daemon actually runs every time the system boots. Otherwise, you might end up scheduling a job, shutting down the system, powering it back on, expecting that job to run at the scheduled time that you set in the future, and it won't because atd daemon didn't get loaded when the system booted up.

So you might want to use the insserv or the chkconfig command, and make sure that the atd daemon is automatically started whenever the system boots.

If your system is based on systemd instead of init, then you start the atd daemon using the command shown right here: systemctl start atd. And you also need to make sure that it starts automatically when the system boots.

You can do that with that systemctl command as well; just use the enable parameter.

Restrict Access to atd 2:09-4:12

Once you have your daemon enabled and running, the next thing you should probably do is configure which users on your system are going to be allowed to create at jobs. This can be done by editing the two files that are shown here.

First of all, we have /etc/at.allow and /etc/at.deny. As their names imply, the users in the at.allow file are allowed to create at jobs, while the users in the at.deny file are not allowed to create jobs.

There's a key point here that you have to understand, and it trips up a lot of Linux system administrators because it's kinda counterintuitive. Here's what you need to know. If this file exists, at.allow, then only the users listed in this file are allowed to use at to create at jobs that will run in the future.

If at.allow exists, then this file down here, at.deny, will be ignored. On the other hand, if at.allow is not in the /etc directory, then just the users listed in the at.deny file will be blocked from using at. Everyone else will be allowed to.

If at.allow exists, then only the users in at.allow will be allowed to create at jobs--no one else will. Conversely, if at.allow doesn't exist, and at.deny does, then everyone is allowed to submit at jobs, except for those in the at.deny file.

These are just basic text files. You just open them up in a text editor and enter in the names that you want to either allow or deny. If you make any changes to these files, be aware that the at daemon does not have to be restarted.

These are dynamically read every time a user tries to create a new at job, so you don't have to restart anything. It will just work.

Define a New atd Job 4:13-4:36

At this point, you're ready to use at to schedule a command to run at a future time. The syntax is pretty straightforward.

At the shell prompt, you just type 'at', and then you specify a time when you want the job to run. at is very flexible as to how it allows you to specify when you want these commands to be run.

Use Fixed Time References 4:37-6:12

You can specify either a fixed time reference, or you can specify a relative time reference. A fixed time reference is just what its name implies. It's a specific time in the future not relative to anything else.

One option is to specify the exact hour and minute when you want at to run the commands that you specify. If you use this type of syntax with the at command, the at daemon is going to assume that the hour and the minute you specify is for today unless that time that you specify has already passed, and then it's going to assume that it is tomorrow.

If you want to, you could also add an AM or a PM here, if you're not using a 24-hour clock, in order to specify whether you're referring to 8:00 in the morning or 8:00 in the evening, for example.

There are other keywords that you can use to specify a fixed start time. For example, you can specify noon, midnight, or teatime. Noon specifies 12 PM, midnight specifies 12 AM, and teatime specifies 4 PM.

You can also be really granular about how you specify when a command is to be run. For example, you could specify the exact month, day, and year when you want a command to be run, using any of these three syntaxes.

Or you can specify the exact month, date, year, and time when you want the command to be run. Basically, we're mixing this option and this option into one command. We say on this date, at this time, we want this command to be run.

Use Relative Time References 6:13-7:23

In addition to fixed time references, we can also use relative time references. Relative time references are not fixed at a particular point in time. They're just relative to the current time.

One option would be now, which would run the command now, which really doesn't make a whole lot of sense. Why would you use at? If you wanted to run a command now, you would just run it right now from the shell prompt.

What we usually do, though, is specify now plus a certain value. For example, I could enter 'now +5 minutes', or 'now +2 hours', or 'now +3 days'. I've shown an example of how to do that down here.

I've entered 'at now +10 minutes' to indicate that I want the commands that I specify here to be run 10 minutes in the future of whatever time it currently is. You can also specify today, and that specifies that the command be run today.

You can actually mix this value with a fixed value, such as '2:00 PM today'. Today is the relative time reference; 2:00 PM is the fixed time reference. We could do the very same thing with tomorrow. We could say '2:00 PM tomorrow'.

Add Commands for atd to Run 7:24-10:20

After you enter the at command and a time value, you'll see this prompt displayed. It's the at prompt. At the at prompt, you enter in all of the commands that you want at to run for you at the time specified.

In this example, in 10 minutes I want to run the tail command to look up the boot.log file and write the output to a file named logfile.txt in my /home directory. I also want to use the echo command to write, "The logfile.txt file has been created."

This brings up a very important point that you have to be aware of, and that is the fact that if the commands that you enter at the at prompt will output some type of text on the screen for their standard out after they're done running, you're not going to be able to see that text because at is going to run these jobs in the background.

In this situation, you have two different options for viewing the output. One option is to redirect the output to a file in the file system. That's what I did in this first command.

Normally, you use the tail command to display the text of the specified file on the screen. Well, because we won't be able to see that text when it's run by at, I instead redirect the output from the command to a text file in the file system that I can view later with cat or whatever utility I want.

The other option is to not do anything at all, and that's what I've done on this second command. I just said echo "The logfile.txt file has been created." If I were to enter this at the regular shell prompt, then echo would simply display this text on the screen. But remember, with at, that won't happen. We won't see this text echoed on the screen.

If you don't tell at to do anything otherwise, what it's actually going to do is capture the standard out from the command--whatever would've been written on the screen--and at will actually email you the output to your local user account, which is kind of handy.

That's why I configured this at job in this way. When this job runs, I'm going to get an email that says, "Hey! This file has been created," and then I know I can go look in the file system and access the logfile.txt file to see the output of the first command.

You can add a whole bunch of commands to the at job. Just hit Enter at the end of every single command, and the at prompt will be displayed. You can just keep adding commands until you get everything included in the job that you want run.

Just remember that each command needs to be on its own line. Then, when you're done entering commands, press the Ctrl+D key sequence. When you do, the at prompt will disappear, the job will be scheduled, and a job number will be assigned.

In this case, we have a job number of 5 assigned. Basically, the at daemon will just sit there and look at the clock, and when 10 minutes have passed, it will say, "Oh, I better run job number 5."

View Pending atd Jobs 10:21-11:18

Once you've configured at jobs, you can then use the atq command to view a list of pending jobs.

Be aware that if you're logged in as a regular user--as in the example right here, I'm logged in as the rtracy user--then the atq command will display only the jobs that are associated with my user account.

If you need to see all of the pending jobs for all of the users on the system, then you need to log in as root and run the atq command.

If you create an at job and then realize, "I should not have created that at job, I need to get rid of it because I really don't want it to run," you can remove that pending job from the queue of pending at jobs, using the atrm command.

Just enter 'atrm', followed by the job number. Basically, what you need to do is run atq first, get the job number of the job that you do not want, and then run the atrm command, followed by that job number that you identified.

Summary 11:19-11:27

That's it for this lesson. In this lesson, we reviewed how you can use the atd daemon to schedule jobs to be run in the future. We also discussed how to view pending at jobs, as well as how to remove pending at jobs.
