10.3.5 Scheduling Tasks with cron

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

Schedule Tasks With cron 0:00-0:07

In this demonstration we're going to talk about scheduling tasks using the cron daemon.

Advantages of Using the cron Daemon 0:08-0:55

The at daemon can also be used to schedule tasks in the future. But it has one glaring weakness, and that is the fact that it can run the job that you specify in the future only once, which really doesn't work very well for jobs that have to be run on a regular basis, such as running a backup of your system every night.

The cron daemon works much better in this type of scenario. Using cron, you can create a custom schedule for when the job should be run and then the cron daemon will automatically run that job for you according to the schedule that you specify.

The reason this works is because the cron daemon, which as you can see here is installed and running by default, checks once every minute to see if there are any jobs that it should run, and if there are, it runs them. Using the cron daemon, you can configure either system cron jobs or you can define user-specific cron jobs.

System Jobs 0:56-5:33

We're going to look at creating system cron jobs first. I'm going to use the 'ls' command here to view all the files and directories in the /etc directory that begin with cron.

In order to create a system job, you need to create a script containing all the necessary commands and put them in the appropriate cron directory. Notice that there are four directories here that you can use.

We have /cron.daily, /cron.hourly, /cron.monthly, and /cron.weekly. Any scripts in the /cron.daily directory are run once a day. Any scripts in the /hourly directory will be run once an hour. Any scripts in the /monthly directory will be run once a month. Any scripts in the /weekly directory are run once a week.

For example, notice in the /daily directory there is a script called logrotate. Let's take a look at it. Here you can see that once a day, because it's in the /daily directory, this command is going to be run to rotate our logs.

You're not stuck with just the default scripts that are listed here. You can create your own scripts and then add them to the appropriate directory. For example, if you had a script that you wanted to run every hour, you would drop it into the /cron.hourly directory.

Let's run the 'ls' command again. In addition to these four directories that have a predefined schedule, there's also another directory that you need to be aware of called /cron.d. /cron.d is used for system jobs whose schedule doesn't fit in one of these default directories.

For example, you may need to run a particular job on a schedule that's not hourly, daily, or monthly, but is something else. You can define that in /cron.d.

There's one critical thing you need to remember about /cron.d, and that is the fact that you do not put scripts in /cron.d the way you do these other directories. Instead, you add crontab files.

A crontab file is just a text file that contains a list of commands to be run by cron and each command, along with its associated schedule, is listed on a single line in the file. Let's go ahead and look at one of these that's already been defined in /cron.d.

Let's look at 0hourly. You'll notice right away that 0hourly is not listed in green in the output of the ls command, so we know automatically that that is not a script. It's not an executable file the way these files are. Let's take a look at the 0hourly crontab file.

Notice down here that we have a cron job defined in the 0hourly crontab file. There's a very specific syntax that needs to be followed when you define a crontab file. Each line within the crontab file is made up of specific fields.

This first field is the minutes field that specifies how many minutes past the hour that a particular command should be run. In this case, we're going to run the command specified at one minute after the hour.

The second field is the hour field right here. The hour field specifies the hour of the day when the command should be run.

Be aware that the cron daemon does prefer the 24-hour clock, so you should use a value of 0 to 23 in this field. In this case, we actually have a star character in that field, which means all. Anytime you see a star character in one of these fields, it represents all possible values.

The next field is the day field, which specifies the day of the month that the command should be run. The next field specifies the month of the year when the command should be run. The next field specifies the day of the week when the command should be run.

The next field specifies the name of the user that the command should be run as. In this case, we're going to run the command as root. The last field specifies the exact command to be run. In this example, the run-parts command will be used and what's it going to run? It's going to run all of the scripts in /etc/cron.hourly right there.

In this example, at one minute after the hour of every hour of the day of every day of the month of every month of the year of every day of the week, this command is going to be run, and it's going to be run as the root user.

You can define your own system jobs in a custom crontab file. All you have to do is create the crontab file and drop it in the /cron.d directory, and then the command that you specify will be run according to its associated schedule. Just remember that you have to use the same syntax as is used in this example file right here.

User-specific Jobs 5:34-8:55

In addition to creating system jobs, each user on the system can create their own non-system cron jobs. Each user can create their own individual crontab file that is unique to their account and it's stored in /var/pool/cron instead of here in /etc/cron.d directory.

In order to show you how to do this, I'm going to 'exit' out of my root user account. So I'm back to my rtracy user account, and I'm going to create a crontab file for the rtracy user. I do this by typing 'crontab -e', and I'm going to press the Insert key so I can edit the file.

For this scenario, let's suppose that I want to create a backup of my user's home directory every day at 10:01 PM. In order to do this, I enter 01 for my minutes field, then a tab, then I specify the hour when this job should run.

Again, cron likes the 24-hour clock instead of the 12-hour clock, so to run at 10 PM, I need to specify '22'; 22 minus 12 equals 10, or 10 PM. The job is going to run at 10:01 PM.

Then we have to specify what day of the month we want it to run. We're going to run every day of the month. Then we have to specify which months of the year we want the command to run--all of them. Then we have to specify which days of the week we want it to run--do we want it to run Monday through Friday, or every day?

Let's just run it every day. Then I have to specify the name of the command to run. We're going to create our backup using the tar utility, so I have to specify '/bin/tar'. It is important to note that if you're using the tar command from the shell prompt, you just type tar.

Well, that doesn't work with cron. You have to specify the full path to the command you want to run in the crontab file. If you don't know which directory that is, use the 'which' command to find out that information.

Then I specify that I want to create a new tar file and I want it to be compressed using gzip compression. Then we'll use the 'f' command to specify the name of the backup file we want to create.

I want to put it in '/mnt/private' and I want the name of the file to be 'rtracyhome_bak', and I'll give it a '.tar.gz' extension to indicate that it's a tar archive that's been compressed using gzip.

Then I have to specify what I want backed up. I just want back up my /home directory, '~/'. I'll press Escape, then I'll enter 'exit' to save my changes and 'exit' the editor, and the job is now submitted.

At 10:01 tonight, the tar command will run and create that backup file. We can verify this by entering 'crontab -l'. It will then read the crontab file for my user account and output it here on the screen, so we can see what jobs I have configured and when they're going to run.

If for some reason we decide we don't want to have this cron job running anymore and we need to remove it, one option would be to load crontab -e again and just erase it, or you can simply enter 'crontab -r', and when I do, my entire crontab file is removed.

Be aware before you run crontab -r that if there are other commands within that file that you want to keep, you don't want to use crontab -r, because it deletes the entire file.

You should instead go in and use crontab -e to edit out the specific lines, but if you don't want any cron jobs at all defined for a user account, just use the -r option to delete the crontab file entirely.

Summary 8:56-9:03

That's it for this demonstration. In this demo, we talked about using the cron daemon to schedule jobs, we talked about how to use cron to schedule system jobs, and then we talked about how to use cron to define user-specific jobs.