Server System Management - Linux

Lab 5 – part 2 Scheduling

# 

# Lab 5 part 2: Scheduling

You may not always want to execute a command or script immediately, e.g. so as not to disturb users during working hours. Some scripts have to be run repeatedly. Rather than staying on-site after hours and racking up the overtime or setting your alarm clock at 1 am to log in remotely from home to execute the desired task, Unix provides you with two built-in scheduling systems : CRON & AT.

**Notescr**

While both CRON and AT help you schedule tasks, cron is best suited for repeating tasks. at allows you to schedule a certain event just once.

To use cron for tasks meant to run only for your user profile, add entries to your own user's crontab file. To edit the crontab file enter:

*crontab –e*

Edit the crontab using the format described in the next sections. Save your changes. (Exiting without saving will leave your crontab unchanged.) To display the on-line help describing the format of the crontab file enter:

*man 5 crontab*

Commands that normally run with administrative privileges (i.e. they are generally run using sudo) should be added to the root crontab. To edit the root crontab enter:

*sudo crontab -e*

**Crontab Lines**

Each line has five time-and-date fields, followed by a command, followed by a newline character ('\n'). The fields are separated by spaces. The five time-and-date fields cannot contain spaces. The five time-and-date fields are as follows: minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday).

01 04 1 1 1 /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on January 1st plus every Monday in January.

An asterisk (\*) can be used so that every instance (every hour, every weekday, every month, etc.) of a time period is used.

01 04 \* \* \* /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on every day of every month.

Comma-separated values can be used to run more than one instance of a particular command within a time period. Dash-separated values can be used to run a command continuously.

01,31 04,05 1-15 1,6 \* /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 01 and 31 past the hours of 4:00am and 5:00am on the 1st through the 15th of every January and June.

The "/usr/bin/somedirectory/somecommand" text in the above examples indicates the task which will be run at the specified times. It is recommended that you use the full path to the desired commands as shown in the above examples. Enter which somecommand in the terminal to find the full path to somecommand.

The crontab will begin running as soon as it is properly edited and saved.

You may want to run a script some number of times per time unit. For example if you want to run it every 10 minutes use the following crontab entry (runs on minutes divisible by 10: 0, 10, 20, 30, etc.)

\*/10 \* \* \* \* /usr/bin/somedirectory/somecommand

which is also equivalent to the more cumbersome

0,10,20,30,40,50 \* \* \* \* /usr/bin/somedirectory/somecommand

Cron also offers some special strings, which can be used instead of the five time-and-date fields:

string meaning

@reboot Run once, at startup.

@yearly Run once a year, "0 0 1 1 \*".

@annually (same as @yearly)

@monthly Run once a month, "0 0 1 \* \*".

@weekly Run once a week, "0 0 \* \* 0".

@daily Run once a day, "0 0 \* \* \*".

@midnight (same as @daily)

@hourly Run once an hour, "0 \* \* \* \*".

@reboot /path/to/execuable1

The above example will execute /path/to/executable1 when the system starts.

**Crontab Options**

The -l option causes the current crontab to be displayed on standard output.

The -r option causes the current crontab to be removed.

The -e option is used to edit the current crontab using the editor specified by the EDITOR environment variable.

After you exit from the editor, the modified crontab is checked for errors and, if there are no errors, it is installed automatically. The file is stored in /var/spool/cron/crontabs but should only be edited using the crontab command.

**Allowing/Denying User-Level Cron**

If the /etc/cron.allow file exists, then users must be listed in it in order to be allowed to run the crontab command. If the /etc/cron.allow file does not exist but the /etc/cron.deny file does, then users must not be listed in the /etc/cron.deny file in order to run crontab.

**Note :**

In the case where neither file exists, the default on Debian (but not some other Linux and UNIX systems) is to allow all users to run jobs with crontab.

No cron.allow or cron.deny files exist in a standard Debian install, so all users should have cron available by default, until one of those files is created. If a blank cron.deny file has been created, that will change to the standard behavior users of other operating systems might expect: cron only available to root or users in cron.allow.

**Assignment :**

1. (as root) Create the file /tmp/cron\_artefacts and give it demonic permissions (666). Then create the file /etc/cron.deny. Make sure /etc/cron.deny is readable by others (not just root & root group). Then add your regular user to it. Switch to that user (using the su- command) and try to set up a scheduled task. What happens ?

sudo touch /tmp/cron\_artefacts

sudo chmod 666 /tmp/cron\_artefacts

sudo touch /etc/cron.deny

sudo chmod 644 /etc/cron.deny

sudo vim /etc/cron.deny

crontab -e

A black background with white text

Description automatically generated

Crontab commands are generally stored in the crontab file belonging to your user account (and executed with your user's level of permissions). If you want to regularly run a command requiring administrative permissions, edit the root crontab file:

sudo crontab –e

Depending on the commands being run, you may need to expand the root users PATH variable by putting the following line at the top of the root crontab file:

PATH=/usr/sbin:/usr/bin:/sbin:/bin

crontab -e uses the EDITOR environment variable. To change the editor to your own choice, just set that variable. You may want to set EDITOR in your .bashrc because many commands use this variable. For example, in order to set the editor to be nano add this line to .bashrc:

export EDITOR=nano

It is sensible to test that your cron jobs work as intended. One method for doing this is to set up the job to run a couple of minutes in the future and then check the results before finalising the timing. You may also find it useful to put the commands into script files that log their success or failure, for example:

echo "Nightly Backup Successful: $(date)" >> /tmp/mybackup.log

1. Try this principle now using root crontab : Append the words **CRONTAB on** + **the current time** to the contents of the file /tmp/cron\_artefacts every minute in the future. Check the content of /tmp/cron\_artefacts after a few minutes

Note: for this to work you probably need to get rid of the /etc/cron.deny rule because it specifies you.

sudo crontab -e

\* \* \* \* \* echo "CRONTAB on $(date)" >> /tmp/cron\_artefacts

A screen shot of a computer

Description automatically generated

Two other scheduling systems could be of interest : at and run-parts, which provide other approaches to scheduled tasks. In Debian, run-parts offers simple system-wide directories for running commands hourly, daily, weekly, and monthly.

Scripts to be executed in said times can be placed in /etc/cron.hourly/, /etc/cron.daily/, /etc/cron.weekly/, and /etc/cron.monthly/. All scripts in each directory are run as root, and a specific order to running the scripts can be specified by prefixing the scripts' filenames with numbers (see the man page for run‑parts for more details).

1. Create a simple (executable!) script called RunPartsWasHere that appends the line **RUNPARTS HOURLY on** + **current time** to the file /tmp/cron\_artefacts every hour and put that script in the right directory

touch RunPartsWasHere

chmod +x RunPartsWasHere

Contents of RunPartsWasHere:

#!/bin/bash

# Get the current date and time

current\_time=$(date "+%Y-%m-%d %H:%M:%S")

# Append the line to the specified file

echo "RUNPARTS HOURLY on $current\_time" >> /tmp/cron\_artefacts

sudo mv RunPartsWasHere /etc/cron.hourly/

**Note :**

The crontab files discussed above are user crontabs. Each of the above crontabs is associated with a user, even the root crontab, which is associated with the root user.

There is actually another type of crontab, with syntax as follows:

minute(s) hour(s) day(s)\_of\_month month(s) day(s)\_of\_week user command

Note that the only difference from the syntax of the user crontabs is that the line specifies the user to run the job as.

The first type is as follows. As mentioned above cron uses the run‑parts command and /etc/cron.hourly, /etc/cron.weekly, and /etc/cron.monthly directories. However cron itself is invoked from the /etc/crontab file. This file could be used for other cron commands, but probably shouldn't be. Here's an example line from a fictitious /etc/crontab:

00 01 \* \* \* rusty /home/rusty/rusty-list-files.sh

This would run Rusty's command script as user rusty from his home directory. However, it is not usual to add commands to this file. While an experienced user should know about it, it is not recommended that you add anything to /etc/crontab. Apart from anything else, this could cause a problem if the /etc/crontab file is affected by updates! Rusty could lose his command.

The other type is to be found in the directory /etc/cron.d. This directory can contain crontab files. The directory is often used by packages, and the crontab files allow a user to be associated with the commands in them.

Example: Instead of adding a line to /etc/crontab, which Rusty knows is not a good idea, he might well add a file to the directory /etc/cron.d with the name rusty, containing his cron line above. This would not be affected by updates but is a well known location.

When would you use these alternate crontab locations? Well, on a single user machine or a shared machine such as a school or college server, a user crontab would be the way to go. But in a large IT department, where several people might look after a server, then the directory /etc/cron.d is probably the best place to install crontabs - it's a central point and saves searching for them!

You may not need to look at /etc/crontab or /etc/cron.d, let alone edit them by hand. But an experienced user should perhaps know about them and that the packages that he/she installs may use these locations for their crontabs.

**The at command**

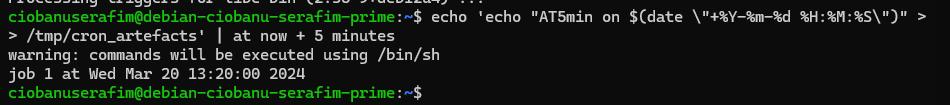
Sometimes you just want to execute a command once, but e.g. outside business hours. Then at is your recommended solution. Install the at package (if it isn’t already). Be amazed at how many required packages are installed alongside it.

Check out the at syntax using man at

1. Using the at command, append the line **AT5min on** + **current time** to the file /tmp/cron\_artefacts 5 minutes from now. What job # is it getting ?

There was no “at” installed. sudo apt install at

echo 'echo "AT5min on $(date \"+%Y-%m-%d %H:%M:%S\")" >> /tmp/cron\_artefacts' | at now + 5 minutes



Number #1

1. Add another at command job to append the line **AT1hour on** + **current time** to the file /tmp/cron\_artefacts 1 hour from now. What job # is that getting ?

echo 'echo "AT1hour on $(date \"+%Y-%m-%d %H:%M:%S\")" >> /tmp/cron\_artefacts' | at now + 1 hour

A black screen with white text

Description automatically generated

1. Show all jobs that are scheduled using atq. What happens after the first job is executed ?

atq

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

The job is getting deleted automatically from the list

1. Stop the hourly job from executing. Which command did you use ?

atrm 2

**Note :**

There is a similar setup to cron allowing / disallowing users to schedule one-shot jobs using at. You find this configuration in the /etc/at.deny and /etc/at.allow files.

1. Prevent your regular user from using at to schedule commands.

A screen shot of a computer

Description automatically generated

1. See if you remember a previous lab : What is the email address of the maintainer of **at** ? (QUIZ)

apt-cache show at | grep Maintainer

[calhariz@debian.org](mailto:calhariz@debian.org)

1. You can do fun things with cron and at. For instance schedule a job that sends a mail to your boss at 11 pm sharp on every workday (from Monday through Friday), with body saying you’re working overtime again in the lab to solve a complicated server configuration issue and subject “Paid Overtime request” to your boss : [my\_boss@gullible.com](mailto:my_boss@gullible.com) (QUIZ)

0 23 \* \* 1-5 echo “I’m working overtime AGAIN to solve this complicated server configuration issue” | mail -S “Paid Overtime request” my\_boss@gullible.com