Chapter 12:
System logging and scheduling tasks

Learning objectives

Upon completing this chapter, the learner should be able to:

- Configure logging
- Read how the rsyslogd and journald services are used to keep log information
- Manage logs that are written by these services
- Configure log rotation and make the journal persistent
- Schedule jobs for future execution
- Configure cron to execute jobs repeatedly at a specific time
- Know that different methods exist to tell cron when a job should be executed
- Learn about the anacron service
- Learn how to use the atd service to schedule tasks to be executed once

Key terms

Log

Log file

rsyslogd

Log priority

debug

info

notice

warning / warn

err / error

crit

alert

emerg / panic

journald

atd

log rotation

anacron

Table of content

1. Understanding system logging

Most services used on a Linux server write information to log files. This information can be written to different destinations, and there are multiple solutions to find the relevant information in system logs. No less than three different approaches can be used by services to write log information:

- **Direct write**: Some services write logging information directly to the log files, even some important services such as the Apache web server and the Samba file server
- **rsyslogd**: it is the enhancement of syslogd, a service that takes care of managing centralized log files. Syslogd has been around for a long time
- **journald**: This service is tightly integrated with systemd, which allows administrators to read detailed information from the journal while monitoring service status using the systemctl status command

1.1. Reading log files

Apart from the messages that are written by journald to the journal, and which can be read using the journalctl command, on a Linux system you'll also find different log files in the directory /var/log. These files can be read using a pager utility like less.

The exact number of files in the /var/log directory will change, depending on the configuration of a server and the services that are running on that server. Some files, however, do exist on most occasions, and as an administrator, you should know which files they are and what kind of contents can be expected in these files. The table below provides an overview of some of the standard files that are created in this directory.

Log File	Explanation
/var/log/messages	The most commonly used log file, it is the generic log file where most messages are written to.
/var/log/dmesg	Contains kernel log messages.
/var/log/secure	Contains authentication related messages. Look here to see which authentication errors have occurred on a server.
/var/log/boot.log	Look here for messages that are related to system startup
/var/log/audit/audit.log	Contains audit messages. SELinux writes to this file.
/var/log/maillog	Look here for mail-related messages.
/var/log/samba	Provides log files for the Samba service. Notice that Samba by default is not managed through rsyslog, but writes directly to the /var/log directory.
/var/log/sssd	Contains messages that have been written by the sssd service, which plays an important role in the authentication process.
/var/log/cups	Contains log messages that were generated by the print service CUPS.
/var/log/httpd/	Directory that contains log files that are written by the Apache web server. Notice that Apache writes messages to these files directly and not through rsyslog.

1.2. Understanding log file contents

As an administrator, you need to be able to interpret the contents of log files. For example, Listing shows partial content from the /var/log/messages file.

```
[root@server1:~]# tail -n 5 /var/log/messages
Feb  7 16:10:01 server1 systemd: Started Session 145 of
user root.
Feb  7 16:13:04 server1 systemd: Started Session 146 of
user root.
Feb  7 16:13:04 server1 systemd-logind: New session 146
of user root.
Feb  7 16:13:04 server1 dbus[1198]: [system] Activating
service name='org.freedesktop.problems' (using
servicehelper)
Feb  7 16:13:04 server1 dbus[1198]: [system] Successfully
activated service 'org.freedesktop.problems'
```

As you can see in Listing, each line that is logged has specific elements:

- **Date and time:** Every log message starts with a timestamp. For filtering purposes, the timestamp is written as military time
- **Host:** The host the message originated from. This is relevant because rsyslogd can be configured to handle remote logging as well
- **Service or process name:** The name of the service or process that generated the message
- **Message content:** The content of the message, which contains the exact message that has been logged

To read the content of a log file, you can use a pager utility, like less, or you can live monitor what is happening in the log file, as described in the next section.

1.3. Live log file monitoring

When you are configuring services on Linux, it might be useful to see in real time what is happening. You could, for example, open two terminal sessions at the same time. In one terminal session, you configure and test the service. In the other terminal session, you see in real time what is happening.

The tail -f <logfile> command shows in real time which lines are added to the log file, when monitoring a log file with tail –f, the trace remains open until you use Ctrl + C to close it.

1.4. Using logger

Most services write information to the log files all by themselves, where the logger command enables users to write messages to rsyslog from the command line. Just type logger followed by the message you want to write to the logs, so the command offers a convenient solution to write messages from scripts. This allows you to have a script to write to syslog if something goes wrong.

When using logger, you can also specify the priority and facility to log to. The command logger -p kern.err message writes a message to the kernel facility, for example, using the error priority. This option enables you to test the working of specific rsyslog facilities. In the following exercise, you use logger to write log messages.

Exercise: Using live log monitoring and logger

In this exercise, you use tail –f to monitor a log file in real time. You also use logger to write messages to a log file.

1. Open a root shell.
2. From the root shell, type tail –f /var/log/messages.
3. Open a second terminal window. In this terminal window, type su –user to open a subshell as user.
4. Type su –to open a root shell, but enter the wrong password.
5. Notice that nothing appears in /var/log/messages. That is because login–related errors are not written here.
6. From the user shell, type logger hello. You'll see the message appearing in the /var/log/messages file in real time.
7. In the tail –f terminal, use Ctrl+C to stop tracing the messages file.
8. Type tail –n 20 /var/log/secure. This shows the last 20 lines in /var/log/secure, which also shows the messages that the su –password errors have generated previously.

1.5. Rotating log files

To prevent syslog messages from filling up your system completely, the log messages can be rotated. That means that when a certain threshold has been reached, the old log file is closed and a new log file is opened. The logrotate utility is started periodically through the crond service to take care of rotating log files.

When a log file is rotated, the old log file is typically copied to a file that has the rotation date in it. So, if /var/log/messages is rotated on January 17, 2019, the rotated filename will be /var/log/messages-20190115. As a default, four old log files are kept on the system. Files older than that period are removed from the system automatically.

Log files that have been rotated are not stored anywhere, they are just gone. If your company policy requires you to be able to access information about events that have happened more than 5 weeks ago, you should take measures. You could decide either to back up log files or to configure a centralized log server where logrotate keeps rotated messages for a significantly longer period.

The default settings for log rotation are kept in the file /etc/logrotate.conf.

```
[root@server1:~]# cat /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
# use date as a suffix of the rotated file
dateext
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this
directory include /etc/logrotate.d
# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
monthly
create 0664 root utmp
minsize 1M
rotate 1
}
/var/log/btmp {
missingok
monthly
create 0600 root utmp
rotate 1
}
```

The most significant settings used in this configuration file tell logrotate to rotate files on a weekly basis and keep four old versions of the file. If specific files need specific settings, you can create a configuration file for that file in /etc/logrotate.d. The settings for that specific file overwrite the default settings in /etc/logrotate.conf.

2. Understanding the role of rsyslogd and journald

On RHEL7, journald (which is implemented by the system-journald daemon) provides an advanced log management system. journald collects messages from the kernel, the entire boot procedure, and services and writes these messages to an event journal, see Working with journald below.

Because the journal that is written by journald is not persistent between reboots, messages are also forwarded to the rsyslogd service. Rsyslogd writes the messages to different files in the /var/log directory. rsyslogd also offers features that do not exist in journald, such as centralized logging and filtering messages by using modules.

In the current state of RHEL7, journald is not a replacement for rsyslog, it is just another way of logging information. journald is tightly integrated with systemd and therefore logs everything that your server is doing. rsyslogd adds some services to it. In particular, it takes care of writing log information to specific files (that will be persistent between reboots), and it allows you to configure remote logging and log servers.

To get more information about what has been happening on a machine running RHEL, administrators have to take three approaches:

- The files in /var/log that are written by rsyslogd must be monitored
- The journalctl command can be used to get more detailed information from the journal
- For a short overview of the last significant events that have been logged by systemd units through journald, administrators can use the systemctl status <unit> command. This command shows the status of services, as well as the last couple of lines that have been logged

2.1. Configuring rsyslogd

To make sure that the information that needs to be logged is written to the location where you want to find it, you can configure the rsyslogd service through the /etc/rsyslog.conf file. In this file, you find different sections that allow you to specify where and how information should be written.

2.1.1. Understanding rsyslogd Configuration Files

The configuration for rsyslogd is not defined in just one configuration file, the /etc/rsyslogd.conf file is the central location where rsyslogd is configured. From this file, the content of the directory /etc/rsyslog.d is included. This directory can be populated by installing RPM packages on a server. When looking for specific log configuration, make sure to always consider the contents of this directory also.

If specific options need to be passed to the rsyslogd service on startup, you can do this by using the /etc/sysconfig/rsyslog file. This file by default contains one line, which reads SYSLOGD_OTIONS="". On this line, you can specify rsyslogd startup parameters. The SYSLOGD_OPTIONS variable is included in the systemd configuration file that starts rsyslogd. Theoretically, you could change startup parameters in this file, as well, but that is not recommended.

It is important to remember that RHEL7 often has configuration files in two locations. Do not ever change configuration files that are in the /usr/lib directory, only apply modifications to configuration files in the /etc directory.

2.1.2. Understanding rsyslog.conf Sections

The rsyslog.conf file is used to specify what should be logged and where it should be logged. To do this, you'll find different sections in the configuration file:

- #### MODULES ####: rsyslogd is modular. Modules are included to enhance the supported features in rsyslogd.
- #### GLOBAL DIRECTIVES ####: This section is used to specify global parameters, such as the location where auxiliary files are written or the default timestamp format.
- #### RULES ####: This is the most important part of the rsyslog.conf file. It contains the rules that specify what information should be logged to which destination.

Listing shows an example of the RULES section in rsyslog.

```
#### RULES ####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.* /dev/console
# Log anything (except mail) of level info or higher.
# Do not log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages
# The authpriv file has restricted access.
authpriv.* /var/log/secure
# Log all the mail messages in one place.
mail.* -/var/log/maillog
# Log cron stuff
cron.* /var/log/cron
# Everybody gets emergency messages
*.emerg :omusrmsg:*
# Save news errors of level crit and higher in a special
file.
uucp,news.crit /var/log/spooler
```

To determine which types of messages should be logged, different severities can be used in rsyslog.conf lines. These severities are the syslog priorities. Table below provides an overview of the available priorities in ascending order.

Priority	Used for
e	Debug messages that will give as much information as possible about service operation.
info	Informational messages about normal service operation.
notice	Used for informational messages about items that might become an issue later.
warning / warn	Something is suboptimal, but there is no real error yet.
err / error	A noncritical error has occurred.
crit	A critical error has occurred.
alert	Used when the availability of the service is about to be discontinued.
emerg / panic	Message generated when the availability of the service is discontinued.

When a specific priority is used, all messages with that priority and higher are logged according to the specifications used in that specific rule. If you need to configure logging in a detailed way, where messages with different priorities are sent to different files, you can specify the priority with an equals sign (=) in front of it, as in the following configuration file, which will send all cron debug messages to a specific file with the name /var/log/cron.debug. Notice the use of the hyphen (-) in front of the destination filename, which ensures that messages are buffered and not written immediately to disk (which is good for disk performance). For example, cron=debug -/var/log/cron.debug.

Exercise: Changing rsyslog.conf rules

In this exercise, you learn how to change rsyslog.conf. You configure the Apache service to log messages through syslog, and you create a rule that logs debug messages to a specific file.

1. By default, the Apache service does not log through rsyslog, but keeps its own logging. You are going to change that. To start, type yum install -y httpd to install the Apache service.
2. After installing the Apache service, open its configuration file /etc/httpd/conf/httpd.conf and add the following line to it:

   ```
   ErrorLog syslog: local1
   ```
3. Type systemctl restart httpd.
4. Now create a line in the rsyslog.conf file that will send all messages that it receives for facility local1 (which is now used by the httpd service) to the file /var/log/httpd-error.log. To do this, include the following line:

   ```
   local1: error -/var/log/httpd -error.log
   ```
5. Tell rsyslogd to reload its configuration, by using systemctl restart httpd.
6. All Apache error messages will now be written to the httpd -error.log file.
7. From the Firefox browser, go to http://localhost/nowhere. Because the page you are trying to access does not exist, this will be logged to the Apache error log.
8. Now let's create a snap-in file that logs debug messages to a specific file as well. To do this, type:

   ```
   echo "*. debug /var/log/messages-debug" > /etc/rsyslog.d/debug.conf.
   ```
9. Again, restart rsyslogd using systemctl restart rsyslog.
10. Use the command tail -f /var/log/messages-debug to open a trace on the newly created file.
11. Type logger -p daemon. debug "Daemon Debug Message". You'll see the debug message passing by.
12. Use Ctrl+C to close the debug log file.

2.2. Working with journald

The systemd-journald service stores log messages in the journal, a binary file that is stored in the file /run/log/journal. This file can be examined using the journalctl command.

2.2.1. Using journalctl to find events

The easiest way to use journalctl is by just typing the command. It shows you recent events that have been written to the journal since your server last started. Notice that the result of this command is shown in the less pager, and by default you'll see the beginning of the journal. Because the journal is written from the moment your server boots, this is showing boot-related log messages.

If you want to see the last messages that have been logged, you can use journalctl -f, which shows the last lines of the messages where new log lines are automatically added. You can also type journalctl and use (uppercase) G to go to the end of the journal. Also note that the search options / and? work in the journalctl output. Listing shows a partial result of this command.

```
[root@server1:~]# journalctl -f
-- Logs begin at Wed 2020-01-29 22:57:20 EET. --
Feb  08  16:01:01  server1.localdomain.com  CROND[6439]:
(root) CMD (run-parts /etc/cron.hourly)
Feb 08 16:01:01 server1.localdomain.com run-
parts(/etc/cron.hourly)[6442]: starting 0anacron
Feb 08 16:01:02 server1.localdomain.com run-
parts(/etc/cron.hourly)[6448]: finished 0anacron
Feb 08 16:01:02 server1.localdomain.com run-
parts(/etc/cron.hourly)[6450]: starting eachhour
Feb  08  16:01:02  server1.localdomain.com  root[6454]:  This
message is written at Sat Feb 8 16:01:02 EET 2020
Feb 08 16:01:02 server1.localdomain.com run-
parts(/etc/cron.hourly)[6456]: finished eachhour
```

What makes journalctl a flexible command is that its many filtering options allow you to show exactly what you need.

Exercise: Discovering journalctl

In this exercise, you learn how to work with different journalctl options.

1. Type journalctl. You’ll see the content of the journal since your server last started, starting at the beginning of the journal. The content is shown in less, so you can use common less commands to walk through the file.
2. Type q to quit the pager. Now type journalctl −−no−pager. This shows the contents of the journal without using a pager.
3. Type journalctl −f. This opens the live view mode of journalctl, which allows you to see new messages scrolling by in real time. Use Ctrl+C to interrupt.
4. Type journalctl and press the Tab key twice. This shows specific options that can be used for filtering. Type, for instance, journalctl_UID=0.
5. Type journalctl −n 20. The −n 20 option displays the last 20 lines of the journal (just like tail −n 20).
6. Now type journalctl −p err. This command shows errors only.
7. If you want to view journal messages that have been written in a specific time period, you can use the −−since and −−until commands. Both options take the time parameter in the format YYYY−MM−DD hh:mm:ss. Also, you can use yesterday, today, and tomorrow as parameters. So, type journalctl −−since yesterday to show all messages that have been written since yesterday.
8. journalctl allows you to combine different options, as well. So, if you want to show all messages with a priority err that have been written since yesterday, use journalctl −−since yesterday −p err.
9. If you need as much detail as possible, use journalctl −o verbose. This shows different options that are used when writing to the journal.

Many options can be used to tell the journalctl command which specific information you are looking for. Type, for instance, journalctl_SYSTEMD_UNIT=sshd service to show more information about the sshd systemd unit.

In the preceding exercise, you typed journalctl -o verbose to show verbose output. Listing shows an example of the verbose output. You can see that this is providing detailed information for all items that have been logged, including the PID, the ID of the associated user and group account, the command that is associated, and more.

```
[root@server1:~]# journalctl -o verbose
-- Logs begin at Wed 2020-01-29 22:57:20 EET, end at Sat
2020-02-08 16:40:01 EET. --
Wed         2020-01-29         22:57:20.567791         EET
[s=3545349b4d1b4dd5affcfd2cb99d5044;i=1;b=cc8c89c5476e47d
e99f5d22c5647d88c;m=ea3c3;t=59d4d962475ef;x=5549a8572971a
058]
    PRIORITY=6
    _TRANSPORT=driver
    MESSAGE=Runtime  journal  is  using  6.0M  (max  allowed
48.6M, trying to leave 72.9M free of 480.0M available →
current limit 48.6M).
    MESSAGE_ID=ec387f577b844b8fa948f33cad9a75e6
    _PID=90
    _UID=0
    _GID=0
    _COMM=systemd-journal
    _EXE=/usr/lib/systemd/systemd-journald
```

3. Configuring cron to automate recurring tasks

On a Linux system, some tasks have to be automated on a regular basis. It would be one option to configure each process with a process-specific solution to handle recurring tasks, but that would not be efficient to deal with. That is why on Linux the cron service is used as a generic service to run processes automatically at specific times.

The cron service consists of the cron daemon crond, which looks every minute to see whether there is work to do. This work to do is defined in the cron configuration, which consists of multiple files working together to provide the right information to the right service at the right time.

3.1. Managing the cron Service

The cron service is started by default on every RHEL system. The service is needed because some system tasks are running through cron as well. An example of these is logrotate.

Managing the cron service itself is easy: It does not need much management. Where other services need to be reloaded or restarted to activate changes to their configuration, this is not needed by cron. The cron daemon wakes up every minute and checks its configuration to see whether anything needs to be started.

To monitor the current status of the cron service, you can use the systemctl status crond -l command.

```
[root@server1 ~]# systemctl status crond -l
crond.service - Command Scheduler
Loaded:   loaded   (/usr/lib/systemd/system/crond.service;
enabled)
Active:  active  (running)  since  Wed  2019-02-11  03:50:14
EST; 5 days
ago
Main PID: 550 (crond)
CGroup: /system.slice/crond.service
└─550 /usr/sbin/crond -n
```

The most significant part of the output is in the beginning: It mentions that the cron service is loaded and that it is enabled as well. The fact that the service is enabled means that it will automatically be started whenever this service is restarting. The last part of the command shows current status information. Through the journald service, the systemctl command can find out what is actually happening to the crond service.

3.1.1. Understanding cron Timing

When scheduling services through cron, you need to specify when exactly the services need to be started. In the crontab configuration (which is explained more in depth in the next section), you use a time string to indicate when tasks should be started. Table shows the time and date fields used (in the order specified).

Field	Values
minute	0 – 59
hour	0 – 23
day of month	1 – 31
month	1 – 12 (or names which are better avoided)
day of week 0 – 7	(Sunday is 0 or 7, or names [which are better avoided])

In any of these fields, you can use an * to refer to any value. Ranges of numbers are allowed, as are lists and patterns. Some examples are listed next:

- * 11 * * * Any minute between 11:00 and 11:59.
- 0 11 * * 1-5 Every day at 11 a.m. on weekdays only.
- 0 7 - 18 * * 1-5 Every hour on weekdays on the hours from 7 a.m. to 6 p.m.
- 0 */2 2 12 5 Every 2 hours on the hour on December second and every Friday in December.

3.1.2. Managing cron configuration files

The main configuration file for cron is /etc/crontab, but you will not change this file directly. It does give you a convenient overview, though, of the different time specifications that can be used in cron. It also sets environment variables that are used by the commands that are executed through cron.

```
[root@server1 ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# For details see man 4 crontabs
# Example of job definition:
# .---------------- minute (0 - 59)
# | .------------- hour (0 - 23)
# | | .---------- day of month (1 - 31)
# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
```

To make modifications to the cron jobs, there are other locations where cron jobs should be specified:

- Cron files in /etc/cron.d
- Scripts in /etc/cron.hourly, cron.daily, cron.weekly, and cron.monthly
- User-specific files that are created with crontab -e

To create a user specific cron job, type crontab -e after logging in as that user, or as root type crontab -e -u username. When you are using crontab –e, the vi editor opens and creates a temporary file. After you edit the cron configuration, the temporary file is moved to its final location in the directory /var/spool/cron. In this directory, a file is created for each user. These files should never be edited directly! When the file is saved by crontab -e, it is activated automatically.

On RHEL7, if you want to add cron jobs that are not bound to a specific user account (and which for that reason by default will be executed as root if not specified otherwise), you add these to the /etc/cron.d directory. Just put a file in that directory (the exact name does not really matter) and make sure that it meets the syntax of a typical cron job. In listing below you can see an example of the /etc/cron.d/unbound-anchor cron configuration file (which was inserted to the /etc/cron.d directory upon installation of the unbound Domain Name System [DNS] server).

```
[root@server1 ~]# cat /etc/cron.d/unbound-anchor
# Look to see whether the DNSSEC Root key got rolled, if
so check trust
and update
10  3  1  *  *  unbound  /usr/sbin/unbound-anchor  -a
/var/lib/unbound/root.
anchor -c /etc/unbound/icannbundle.pem
```

This example file contains three elements:

1. First there is the time indication, which has the command start at 3:10 a.m. on the first of every month.
2. Then, the configuration indicates that the command has to be started as the unbound user.
3. The last part has the actual command that needs to be started with some arguments specific to the command and show how this command should be used.

The last way to schedule cron jobs is through the following directories: /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, and /etc/cron.monthly. In these directories, you typically find scripts that are put in there from RPM package files. When opening these scripts, notice that no information is included about the time when the command should be executed. That is because the exact time of execution does not really matter. The only thing that does matter is that the job is launched once an hour, day, week, or month.

3.2. Understanding the purpose of anacron

To ensure regular execution of the job, cron uses the anacron service. This service takes care of starting the hourly, daily, weekly, and monthly cron jobs, no matter at which exact time. To determine how this should be done, anacron uses the /etc/anacrontab file. See the listing below.

```
[root@server1:~]# cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the
jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours
only
START_HOURS_RANGE=3-22

#period in days    delay in minutes    job-identifier
command
1         5        cron.daily                nice run-parts
/etc/cron.daily
7         25       cron.weekly               nice run-parts
/etc/cron.weekly
@monthly 45       cron.monthly              nice run-parts
/etc/cron.monthly
[root@server1:~]#
```

In /etc/anacrontab, the jobs to be executed are specified in lines that contain three fields, as shown above:

1. The first field specifies the frequency of job execution, expressed in days.
2. The second column specifies how long anacron waits before executing the job.
3. The last part is the command that should be executed.

3.3. Managing cron security

By default, all users can enter cron jobs. It is possible to limit which user is allowed to schedule cron jobs by using the /etc/cron.allow and /etc/cron.deny configuration files. If the cron.allow file exists, a user must be listed in it to be allowed to use cron. If the /etc/cron.deny file exists, a user must not be listed in it to be allowed to set up cron jobs.

Exercise: Running scheduled tasks through cron

In this exercise, you apply some of the cron basics. You schedule cron jobs using different mechanisms.

1. Open a root shell. Type cat /etc/crontab to get an impression of the contents of the /etc/crontab configuration file.
2. Type crontab –e. This opens an editor interface that by default uses vi as its editor. Add the following line:

   0 2 * * 1–5 logger message from root
3. Use the vi command: wq! to close the editing session and write changes.
4. Use cd /etc/cron. hourly. In this directory, create a script file with the name eachhour that contains the following line:

   logger This message is written at $(date)
5. Use chmod +x eachhour to make the script executable, if you fail to make it executable, it will not work.
6. Now enter the directory /etc/cron. d and in this directory create a file with the name eachhour. Put the following contents in the file:

   11 * * * * root logger This message is written from /etc/cron.d
7. Save the modifications to the configuration file and go work on the next section. (For optimal effect, perform the last part of this exercise after a couple of hours).
8. After a couple of hours, type grep written /var/log/messages and read the messages that have been written which verifies correct cron operations.

3.4. Configuring at to schedule future tasks

Whereas cron is used to schedule jobs that need to be executed on a regular basis, the atd service is available for services that need to be executed only once. On RHEL7, the atd service is available by default, so all that needs to be done is scheduling jobs.

To run a job through the atd service, you would use the at command, followed by the time the job needs to be executed. This can be a specific time, as in at 14:00, but it can also be a time indication like at teatime or at noon. After you type this, the at shell opens. From this shell, you can type several commands that will be executed at the specific time that is mentioned. After entering the commands, use Ctrl+D to quit the at shell.

After scheduling jobs with at, you can use the atq command (q for queue) to get an overview of all jobs currently scheduled. It is also possible to remove current at jobs. To do this, use the atrm command, optionally followed by the number of the at job that you want to remove.

Exercise: Scheduling jobs with at

In this exercise, you learn how to schedule jobs using the atd service.

1. Type systemctl status at. In the line that starts with Loaded, this command should show you that the service is currently loaded and enabled, which means that it is ready to start receiving jobs.
2. Type at 15:00 (or replace with any time near to the time at which you are working on this exercise).
3. Type logger message from at. Use Ctrl+D to close the at shell.
4. Type atq to verify that the job has indeed been scheduled.

Quiz

Chapter review questions

1. Which of the following statements about journald is not true?

   a. journald logs kernel messages.

   b. journald writes to the journal, which by default does not persist between boots.

   c. journald is a replacement of rsyslogd.

   d. To read files from the journal, the journalctl command is used.

2. Which log would you read to find messages related to authentication errors?

   a. /var/log/messages

   a. /var/log/lastlog

   b. /var/log/audit/audit.log

   c. /var/log/secure

3. Which log would you read to find information that relates to SELinux events?

   a. /var/log/messages

   b. /var/log/lastlog

   c. /var/log/audit/audit.log

   d. /var/log/secure

4. What is the name of the rsyslogd configuration file?

   a. /etc/rsyslog.conf

   b. /etc/sysconfig/rsyslogd.conf

   c. /etc/sysconfig/rsyslog.conf

   d. /etc/rsyslog.d/rsyslogd.conf

5. You need to change the startup behavior of the rsyslogd service. Which of the following describes the recommended approach to do so?
   a. Include the startup parameter in the main rsyslog configuration file.
   b. Create a snap-in file in the directory /etc/rsyslog.d and specify the required parameters in there.
   c. Change the systemd unit file in /usr/lib/systemd/system to include the required startup parameter.
   d. Use the SYSLOGD_OPTIONS line in the file /etc/sysconfig/rsyslog and include the startup parameter here.
6. Which directory is used to store the journald journal?
   a. /var/log/journal
   b. /var/run/journal
   c. /run/log
   d. /run/log/journal
7. What do you need to do to make the journald journal persistent?
   a. Create the directory /var/log/journal, set appropriate permissions and reboot your machine.
   b. Open /etc/sysconfig/journal and set the PERSISTENT option to yes.
   c. Open the /etc/systemd/journald. conf file and set the PERSISTENT option to yes.
   d. Create the /var/log/journal file and set appropriate permissions.
8. Which of the following commands enables you to check the current status of the crond service?
   a. service crond status
   b. systemctl status crond
   c. /usr/sbin/crond–status
   d. Chkconfig crond–show

9. Which of the following would run a cron task Sunday at 11 a.m.?

a. 11 7 * *

b. 0 11 * 7 *

c. 0 11 * * 7

d. 11 0 * 7 *

10. Which of the following launches a job every five minutes from Monday through Friday?

a. */5 * * * 1-5

b. */5 * 1-5 * *

c. 0/5 * * * 1-5

d. 0/5 * 1-5 * *

11. How do you create a cron job for a specific user?

a. Log in as that user and type crontab -e to open the cron editor.

b. Open the crontab file in the user home directory and add what you want to add.

c. As root, type crontab -e username.

d. As root, type crontab -u username -e.

12. Which directory is mainly used by cron files that are installed automatically through RPM?

a. /etc/crond.d.

b. /etc/cron.d.

c. /var/cron.

d. /var/spool/cron.

13. Which of the following is not a recommended way to specify jobs that should be executed with cron?

a. Modify /etc/crontab.

b. Put the jobs in separate scripts in /etc/cron.d.

c. Use crontab -e to create user specific cron jobs.

d. Put scripts in /etc/cron. {hourly|daily|weekly|monthly} for automatic execution.

14. Which service takes care of executing cron jobs in /etc/cron. hourly, cron. daily, cron. weekly, and cron. monthly?

a. cron

b. crontab

c. atd

d. anacron

15. Which of the statements about cron security is true?

a. By default, all users are allowed to schedule tasks through cron because the /etc/cron. allow file has the keyword all in it.

b. If the cron. deny file exists, a cron. allow file must be created also and list users who are allowed to schedule tasks through cron.

c. For every user, a matching entry must exist in either the cron. allow file, or in the cron. deny file.

d. If the cron. allow file exists, a user must be listed in it to be able to schedule jobs through cron.

16. After entering commands in the at shell, which command enables you to close the at shell?

a. Ctrl+V

b. Ctrl+D

c. exit

d. :wq

17. Which command enables you to see current at jobs scheduled for execution?

a. atrm

b. atls

c. atq

d. at

18. Which facility is the best solution if you want to configure Apache to log messages through rsyslog?

a. daemon

b. apache

c. syslog

d. Local 0 − 7

19. Which command enables you to schedule a cron job for user lisa?

a. crontab −u lisa –e

b. crontab lisa –e

c. anacron −u lisa –e

d. cron −u lisa

20. How do you specify that user boris is never allowed to schedule jobs through cron?

a. By default, all users are not allowed to schedule tasks through cron

b. crontab boris –e

c. add boris to the file “cron. deny”

d. add boris to the file “cron. allow”

Answers to chapter Review Questions:

1. c
2. d
3. a
4. a
5. d
6. d
7. a
8. b
9. c
10. a
11. a, d
12. b
13. a
14. d
15. d
16. b
17. c
18. d
19. a
20. c