RHCSA - EX200

Trainer

Ali Aydemir
CISCO, CCIE #47287 SP/RS, CCSI#35413
AVAYA, ACE-Fx #169
HUAWEI, HCDP

RHCSA Timetable

Day	AM	Lunch	PM
1	-Installing RHEL Server -Using Essential Tools	--	-Essential File Management -ToolsWorking with Text Files -Connecting to a RHEL Server
2	-User and Group Management -Permissions Management	--	-Configuring Networking -Process Management -Working with Virtual Machines
3	-Installing Software Packages -Scheduling Tasks	--	-Configuring Logging -Managing Partitions -Managing LVM Logical Volumes
4	-Basic Kernel Management -Configuring a Basic Apache Server	--	-Managing and Understanding the Boot Procedure -Essential Boot Procedure Troubleshooting
5	-Managing SELinux -Configuring a Firewall	--	-Configuring Remote Mounts and FTP -Configuring Time Services

Chapter 11:

Installing Software Packages

Chapter 11 Objectives

- The following topics are covered in this chapter:
  - Managing Software with yum
  - Using yum
  - Managing Software Packages with RPM

- The following RHCSA exam objectives are covered in this chapter:
  - Install and update software packages from Red Hat Network, a remote repository, or from the local file system

Managing Software

Managing software packages is an important task for an administrator of Red Hat Enterprise Linux. In this chapter, you learn how to manage software packages from the command line by using the yum utility. You also learn which role repositories play in software management with yum. In the second part of this chapter, you learn how to manage software with the **rpm** command, which is particularly important to query new and installed software packages.

Managing Software Packages with *yum*

The default utility used to manage software packages on Red Hat Enterprise Linux is yum, which stands for Yellowdog update manager. Yum is designed to work with repositories, which are online depots of available software packages. In this section, you learn how to create and manage repositories and how to manage software packages based on the contents of the repositories.

Understanding the Role of Repositories

Software on Red Hat Enterprise Linux is provided in the RPM (Red Hat Package Manager) format. This is a specific format used to archive the package and provide package metadata as well.

When you are working with software in Red Hat Enterprise Linux, repositories play a key role. Working with repositories makes it easy to keep your server current: The maintainer of the repository publishes updated packages in the repository, and the result is that whenever you use yum (discussed later in this chapter) to install software, the most recent version of the software is automatically used.

Another major benefit of working with yum is the way that package dependencies are dealt with. On Linux (as on most other modern operating systems) software packages have dependencies. This means that to install one package, other packages have to be present as well. Without using repositories, that would mean that these packages have to be installed manually.

Understanding the Role of Repositories

While installing Red Hat Enterprise Linux, it asks you to register with Red Hat Network (RHN). From RHN, different repositories are provided. After registering with RHN, you can install software packages that are verified by Red Hat automatically. If you are using CentOS, you get access to the CentOS repositories. If you choose to install Red Hat Enterprise Linux without a registration key, however, it cannot get in touch with the RHN repositories, and you end up with no repositories at all. In that case, you have to be able to specify yourself which repository you want to use.

Note that repositories are specific to an operating system. Therefore, if you are using RHEL, you should use RHEL repositories only. Do not try, for instance, to add CentOS repositories to an RHEL server. If you want to provide additional software from the Fedora project to an RHEL server (which for support reasons is not recommended), you can consider adding the EPEL (Extra Packages for Enterprise Linux) repositories. See https://fedoraproject.org/wiki/EPEL for more information.

WARNING Do not install EPEL repositories on RHEL; you break your support if you do.

Specifying Which Repository to Use

On most occasions, after the installation of your server has finished, it is configured with a list of repositories that should be used. You sometimes have to tell your server which repositories should be used:

- You want to distribute nondefault software packages through repositories.
- You are installing Red Hat Enterprise Linux without registering it on RHN.

To tell your server which repository to use, you need to create a file with a name that ends in .repo. In that file you need the following contents:

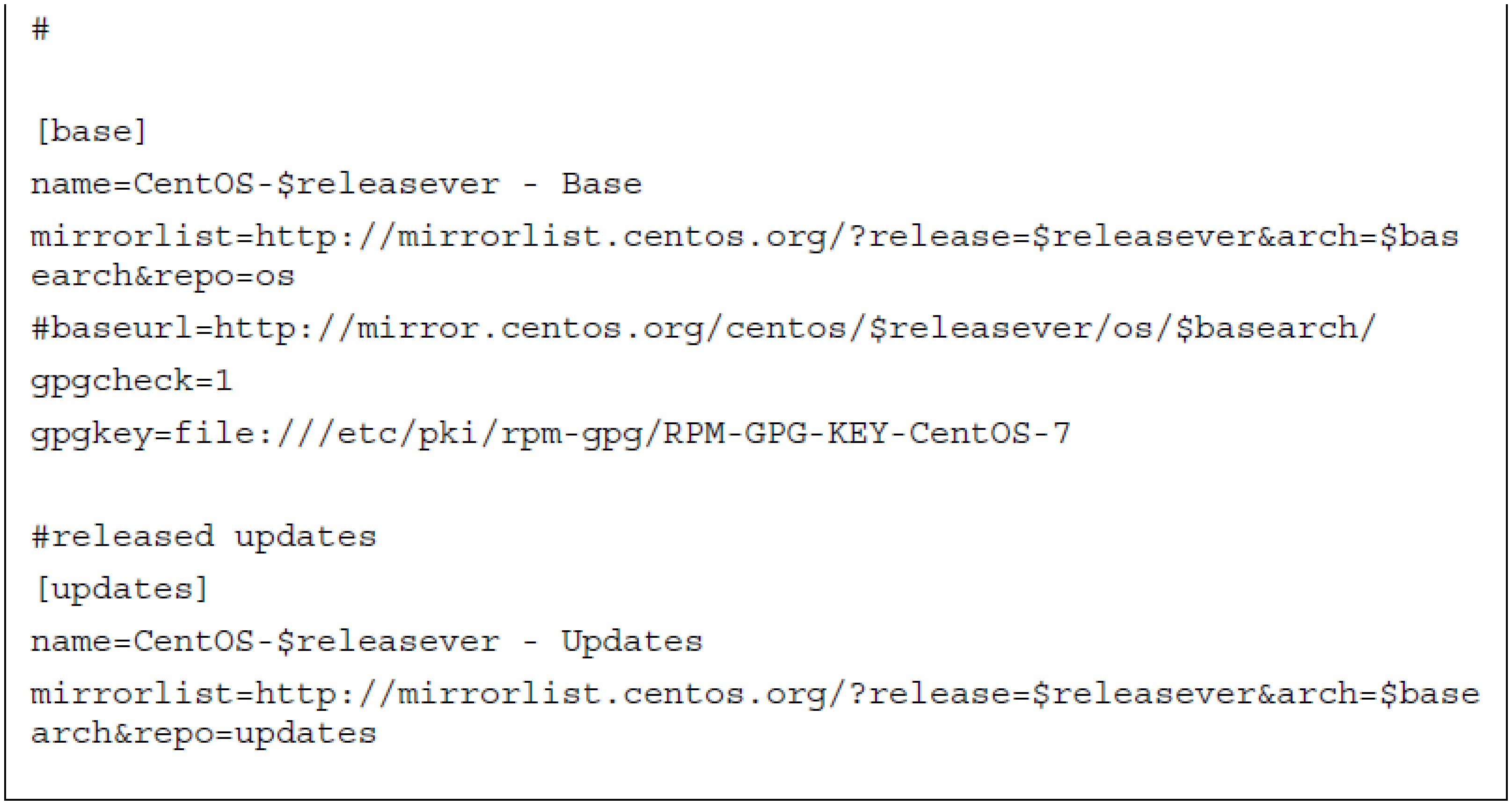
- **[label]** The .repo file can contain different repositories, each section starting with a label that identifies the specific repository.
- **name=** Use this to specify the name of the repository you want to use.
- **baseurl=** Contains the URL that points to the specific repository location.
- In the repository files that are provided by default, you may find several repositories in one file, as is the case in Listing 11.1. This is useful to group repositories that belong together in one file, and is often done in repository files that are provided as a default. If you are creating repository files yourself, you are free to create separate files for each repository.

Specifying Which Repository to Use

Listing 11.1 Repository File Example

```
[root@server1 yum.repos.d]# cat CentOS-Base.repo
# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and
the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client.  You should use this for CentOS
updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try
the
# remarked out baseurl= line instead.
#
#
```

Specifying Which Repository to Use



```
#

[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates
```

Specifying Which Repository to Use

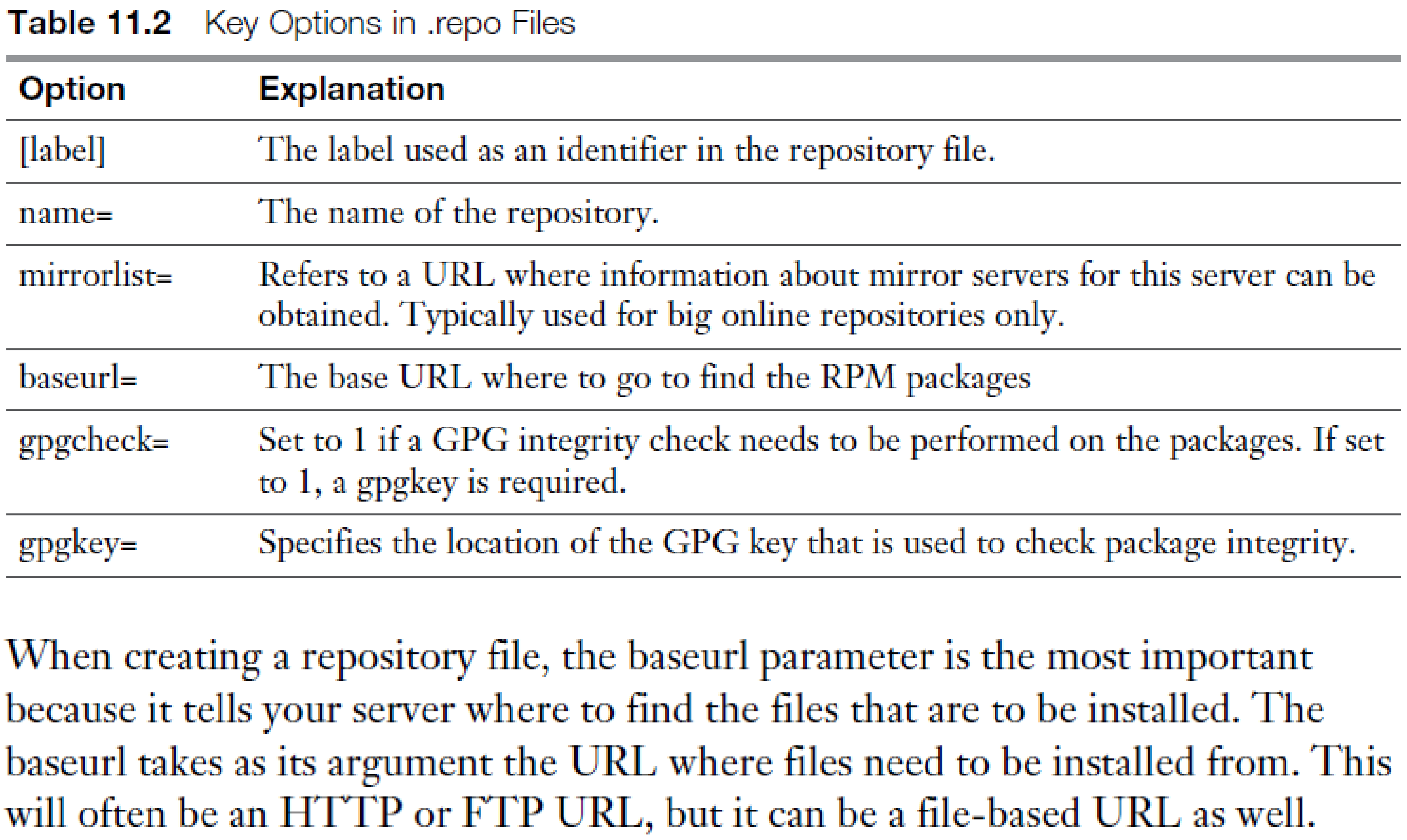


Table 11.2 Key Options in .repo Files

Option	Explanation
[label]	The label used as an identifier in the repository file.
name=	The name of the repository.
mirrorlist=	Refers to a URL where information about mirror servers for this server can be obtained. Typically used for big online repositories only.
baseurl=	The base URL where to go to find the RPM packages
gpgcheck=	Set to 1 if a GPG integrity check needs to be performed on the packages. If set to 1, a gpgkey is required.
gpgkey=	Specifies the location of the GPG key that is used to check package integrity.

When creating a repository file, the baseurl parameter is the most important because it tells your server where to find the files that are to be installed. The baseurl takes as its argument the URL where files need to be installed from. This will often be an HTTP or FTP URL, but it can be a file-based URL as well.

Specifying Which Repository to Use

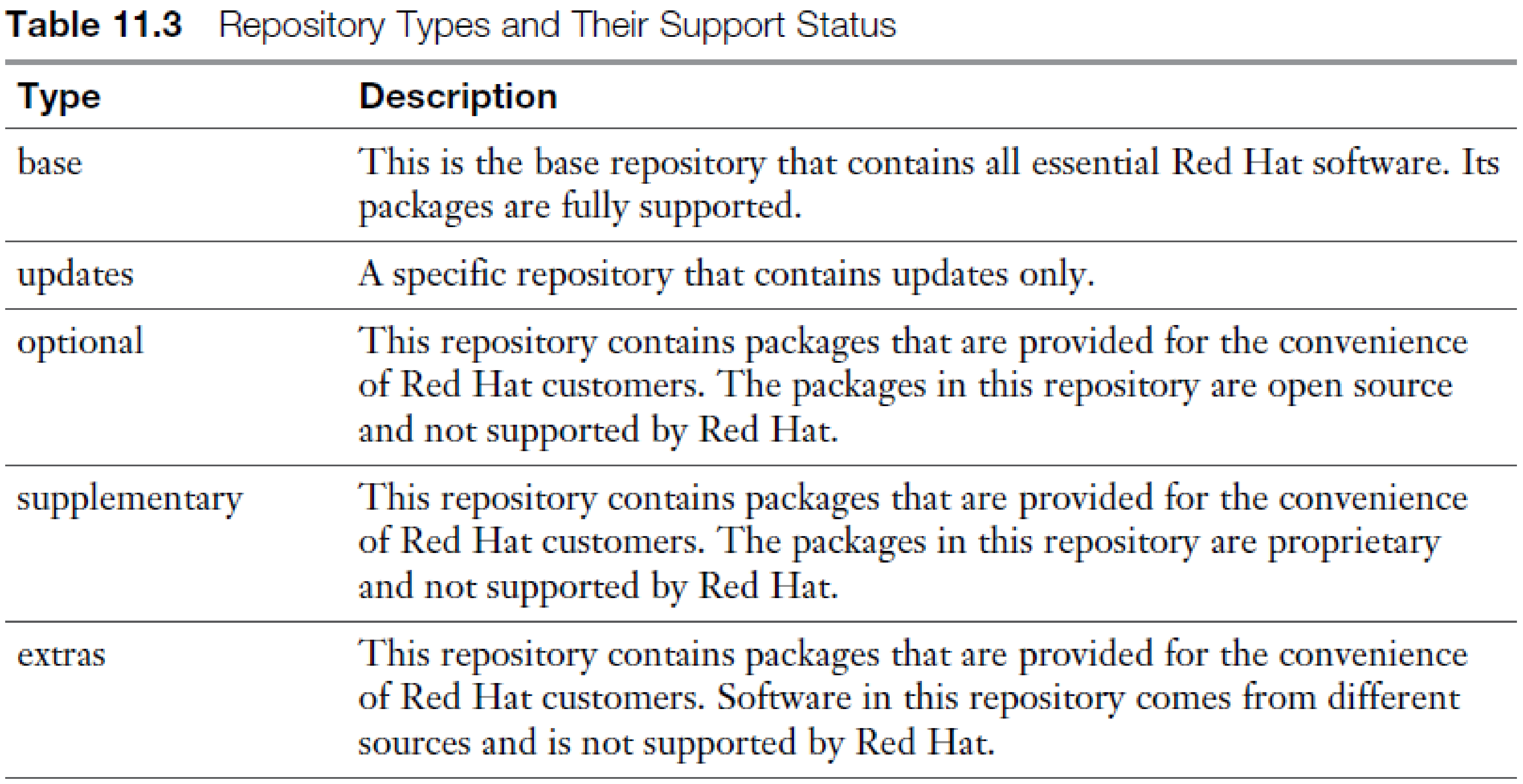


Table 11.3 Repository Types and Their Support Status

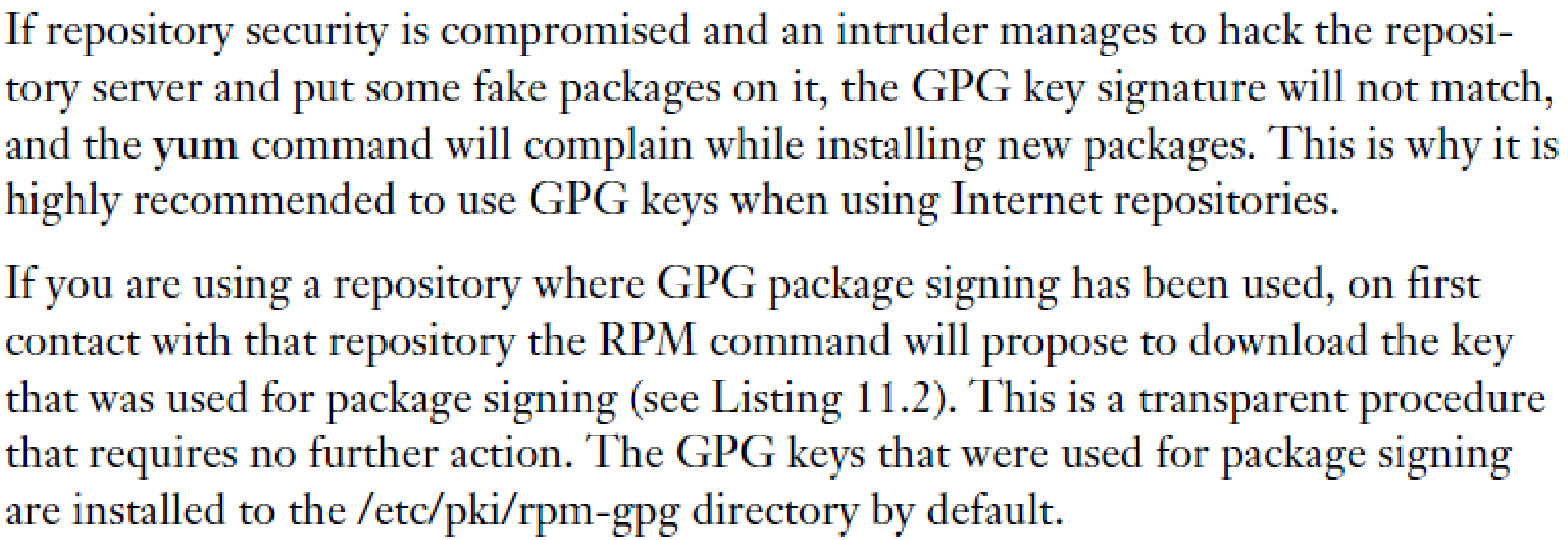
Type	Description
base	This is the base repository that contains all essential Red Hat software. Its packages are fully supported.
updates	A specific repository that contains updates only.
optional	This repository contains packages that are provided for the convenience of Red Hat customers. The packages in this repository are open source and not supported by Red Hat.
supplementary	This repository contains packages that are provided for the convenience of Red Hat customers. The packages in this repository are proprietary and not supported by Red Hat.
extras	This repository contains packages that are provided for the convenience of Red Hat customers. Software in this repository comes from different sources and is not supported by Red Hat.

Understanding Repository Security

Using repositories allows you to transparently install software packages from the Internet. This is convenient, but it also involves a security risk. When installing RPM packages, you do that with root permissions, and if in the RPM package script code is executed, that is executed as root as well. For that reason, you want to make sure that you can trust the software packages you are trying to install. This is why repositories in general use keys for package signing. This is also why on Red Hat Enterprise Linux it is a good idea to use repositories provided though RHN only.

To secure packages in a repository, these packages are often signed with a GPG key. This makes it possible to check whether packages have been changed since the owner of the repository provided them. The GPG key used to sign the software packages is typically made available through the repository as well. The users of the repository can download that key and store it locally so that the package signature check can be performed automatically each time a package is downloaded from the repository.

Understanding Repository Security



If repository security is compromised and an intruder manages to hack the repository server and put some fake packages on it, the GPG key signature will not match, and the **yum** command will complain while installing new packages. This is why it is highly recommended to use GPG keys when using Internet repositories.

If you are using a repository where GPG package signing has been used, on first contact with that repository the RPM command will propose to download the key that was used for package signing (see Listing 11.2). This is a transparent procedure that requires no further action. The GPG keys that were used for package signing are installed to the /etc/pki/rpm-gpg directory by default.

Understanding Repository Security

Listing 11.2 On First Contact with a Repository, the GPG Key Is Downloaded

```
[root@server1 ~]# yum install kernel
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror1.spango.com
 * extras: mirror.netrouting.net
 * updates: mirrors.supportex.net
Resolving Dependencies
--> Running transaction check
---> Package kernel.x86_64 0:3.10.0-229.1.2.el7 will be installed
--> Processing Dependency: linux-firmware >= 20140911 for package:
kernel-3.10.0-229.1.2.el7.x86_64
--> Running transaction check
---> Package linux-firmware.noarch 0:20140213-0.3.git4164c23.el7 will
be updated
```

Creating Your Own Repository

It is not a requirement for the RHCSA or RHCE exam, but it can be useful to know how to create your own repository if you want to test setting up and working with repositories. This allows you to put your own RPMs in a directory and publish that directory as a repository. It is also useful to know how to do this if you have installed RHEL and not connected it to RHN, which means that you would not have any repositories at all.

The procedure itself is not hard to summarize. You need to copy all RPM packages to a directory that you want to use as a repository, and after doing that, you need to use the **createrepo** command to generate the metadata that enables you to use that directory as a repository. Exercise 11.1 describes how to do this.

Exercise 11.1 Creating Your Own Repository

In this exercise, you learn how to create your own repository. To perform this exercise, you need to have access to the CentOS installation disk or ISO file.

1. Insert the installation disk in your virtual machine. This mounts it on the directory /run/media/user/CentOS 7 x86_64. Alternatively, you can manually mount the ISO on the /mnt directory, using **mount -o loop /path/to/centos.iso /mnt**.
2. Type **mkdir /repo** to create a directory /repo that can be used as repository.
3. If you want to create a complete repository, containing all the required files, type **cp $MOUNTPATH/Packages/* repo**. (Replace $MOUNTPATH with the name of the directory on which the installation disk is mounted.) If you do not need a complete repository, you can copy just a few files from the installation disk to the /repo directory.

Exercise 11.1 Creating Your Own Repository

4. Type **yum install -y createrepo** to ensure that the createrepo RPM package is installed.
5. Type **createrepo /repo**. This generates the repository metadata, which allows you to use your own repository.
6. Now that you have created your own repository, you might as well start using it. In the /etc/yum.repos.d directory, create a file with the name my.repo. Make sure this file has the following contents:

   ```
   [myrepo]
   name=myrepo
   baseurl=file:///repo
   ```
7. Type **yum repolist** to verify the availability of the newly created repository. It should show the name of the myrepo repository, including the number of packages that is offered through this repository (see Listing 11.3).

Exercise 11.1 Creating Your Own Repository

Listing 11.3 Verifying Repository Availability with **yum repolist**

```
[root@server1 Packages]# yum repolist
Loaded plugins: fastestmirror, langpacks
myrepo                                                | 2.9 kB     00:00
myrepo/primary_db                                     |  77 kB   00:00
Loading mirror speeds from cached hostfile
 * base: mirror.nl.leaseweb.net
 * extras: mirror.nl.leaseweb.net
 * updates: mirror.nl.leaseweb.net
repo id                    repo name                              status
base/7/x86_64              CentOS-7 - Base                  8,465
extras/7/x86_64            CentOS-7 - Extras                  104
myrepo                     myrepo                             130
updates/7/x86_64           CentOS-7 - Updates               1,668
repolist: 10,367
```

Using *yum*

At this point, you should have operational repositories, so it is time to start using them. To use repositories, you need the **yum** command. This command enables you to perform several tasks on the repositories. Table 11.4 provides an overview of common yum tasks.

Using *yum*

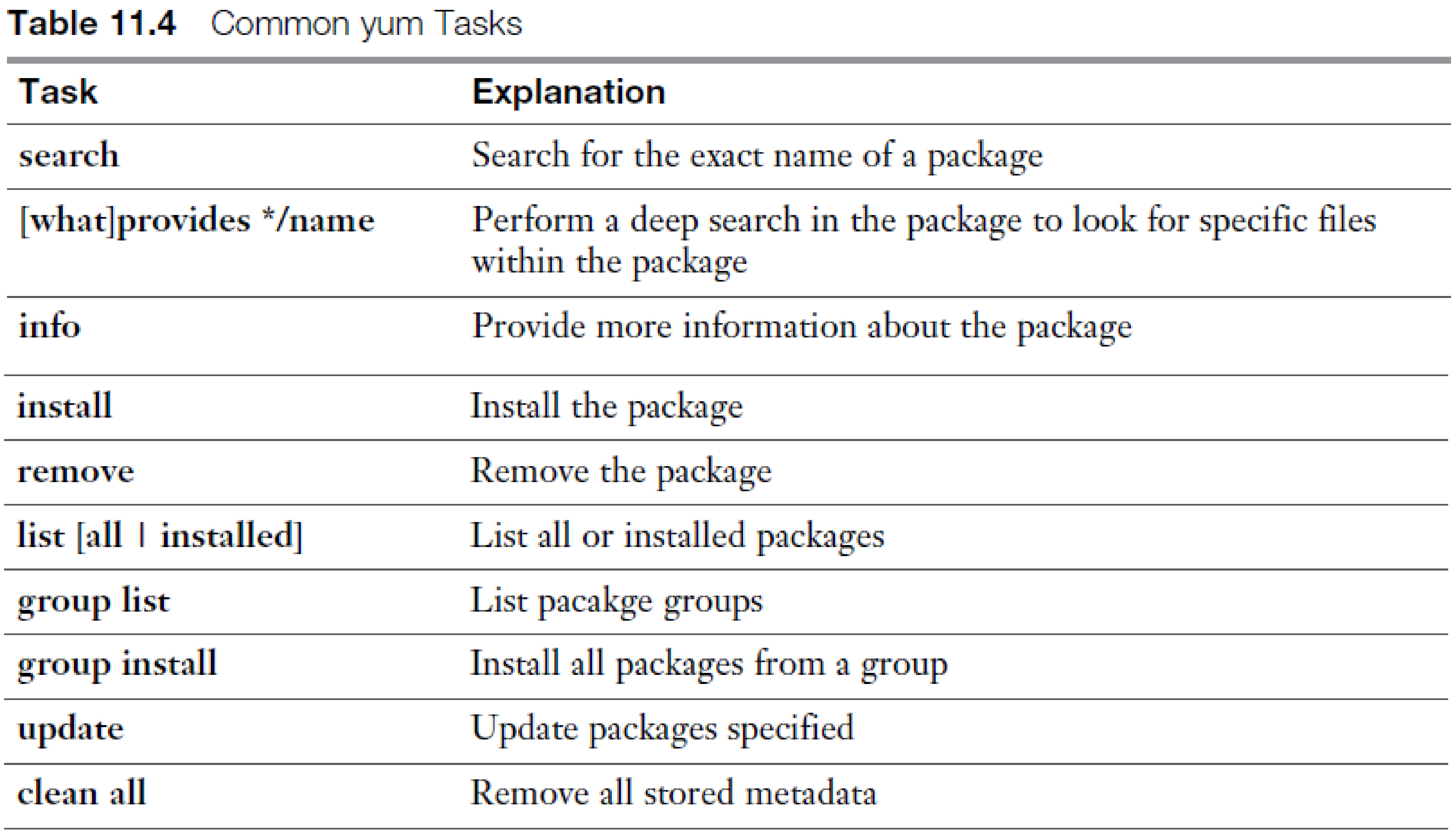


Table 11.4 Common yum Tasks

Task	Explanation
search	Search for the exact name of a package
[what]provides */name	Perform a deep search in the package to look for specific files within the package
info	Provide more information about the package
install	Install the package
remove	Remove the package
list [all | installed]	List all or installed packages
group list	List pacakge groups
group install	Install all packages from a group
update	Update packages specified
clean all	Remove all stored metadata

Using *yum* to Find Software Packages

To install packages with yum, you first need to know the exact name of the package. The **yum search** command can help you with that. When you use **yum search**, it first gets in touch with the online repositories (which might take a minute), after which it downloads the most recent repository metadata to the local machine. Then, **yum search** looks in the package name and description for the string you have been looking for. In Listing 11.4, you can see what the result looks like after using **yum search user**.

Using *yum* to Find Software Packages

Listing 11.4 yum search Sample Output

```
[root@server1 Packages]# yum search user
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.sitbv.nl
 * extras: mirror.sitbv.nl
 * updates: mirror.sitbv.nl
============================== N/S matched: user
===============================
gnome-user-docs.noarch : GNOME User Documentation
libuser.i686 : A user and group account administration library
libuser.x86_64 : A user and group account administration library
libuser-devel.i686 : Files needed for developing applications which use
libuser
libuser-devel.x86_64 : Files needed for developing applications which
use
                     : libuser
```

Using *yum* to Find Software Packages

Because the **yum search** command looks in the package name and summary only, it often does not show what you need. You often need to look for packages containing a specific file. To do this, the **yum whatprovides** or **yum provides** command will help you. (There is no functional difference between these two commands.) To make it clear that you are looking for packages containing a specific file, you need to specify the filename as */filename, or use the full path name to the file you want to use. So if you need to look for the package containing the file semanage, for example, use **yum whatprovides */semanage**. It will show the name of the package as a result.

Getting More Information About Packages

Before installing a package, it is a good idea to get some more information about the package. Because the **yum** command was developed to be intuitive, it is almost possible to guess how that works. Just use **yum info**, followed by the name of the package. In Listing 11.5, you see what this looks like for the nmap package (which, by the way, is a very useful tool). It is a network sniffer that allows you to find ports that are open on other hosts. Just use **nmap 192.168.122.100** to give it a try, but be aware that some network administrators really do not like nmap and might consider this a hostile attack.

Getting More Information About Packages

Listing 11.5 Example Output of **yum info nmap**

```
[root@localhost ~]# yum info nmap
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror.triple-it.nl
 * extras: centos.mirror.triple-it.nl
 * updates: centos.mirror.triple-it.nl
Available Packages
Name        : nmap
Arch        : x86_64
Epoch       : 2
Version     : 6.40
Release     : 4.el7
Size        : 3.9 M
Repo        : base/7/x86_64
Summary     : Network exploration tool and security scanner
URL         : http://nmap.org/
License     : GPLv2 and LGPLv2+ and GPLv2+ and BSD
```

Installing and Removing Software Packages

If after looking at the **yum info** output you are happy with the package, the next step is to install it. As anything else you are doing with yum, it is not hard to guess how to do that: Just use **yum install nmap**. When used in this way, the **yum** command asks for confirmation. If when you type the **yum install** command you are sure about what you are doing, you might as well use the **-y** option, which passes a "yes" to the confirmation prompt that yum normally issues. Listing 11.6 shows what the result looks like.

Listing 11.6 Installing Software with yum

```
[root@localhost ~]# yum install nmap
Loaded plugins: fastestmirror
base
3.6 kB  00:00:00
extras
| 3.4 kB  00:00:00
updates
| 3.4 kB  00:00:00
Loading mirror speeds from cached hostfile
```

Installing and Removing Software Packages

To remove software packages from a machine, use the **yum remove** command. This command also will do a dependency analysis, which means that it will not only remove the selected package but also all packages that depend on it. This may sometimes lead to a long list of software packages that are going to be removed. To avoid unpleasant surprises, you should never use **yum remove** with the **-y** option.

NOTE Some packages are protected. Therefore, you cannot easily remove them. If **yum remove** encounters protected packages, it refuses to remove them.

Showing Lists of Packages

When working with yum, you may also use the **yum list** command to show lists of packages. Used without arguments, **yum list** shows a list of all software packages that are available, including the repository they were installed from. You see the repository names as listed in Table 11.2 and the @anaconda repository as well. If a repository name is shown, the package is available in that specific repository. If @ anaconda is listed, the package has already been installed on this system. Listing 11.7 shows the partial output of the **yum list** command.

Showing Lists of Packages

Listing 11.7 Partial Output of the **yum list** Command

```
Installed Packages
...
zenity.x86_64                  3.8.0-4.el7           @anaconda
zip.x86_64                     3.0-10.el7            @anaconda
zlib.x86_64                    1.2.7-13.el7          @anaconda
Available Packages
389-ds-base.x86_64             1.3.3.1-16.el7_1      updates
389-ds-base-devel.x86_64       1.3.3.1-16.el7_1      updates
389-ds-base-libs.x86_64        1.3.3.1-16.el7_1      updates
Cython.x86_64                  0.19-3.el7            base
ElectricFence.i686             2.2.2-39.el7          base
ElectricFence.x86_64           2.2.2-39.el7          base
GConf2.i686                    3.2.6-8.el7           base
```

Showing Lists of Packages

If you want to see which packages are installed on your server, you can use the **yum list installed** command. The **yum list** command can also prove useful when used with the name of a specific package as its argument. For instance, type **yum list kernel** to show which version of the kernel is actually installed and which version is available as the most recent version in the repositories. Listing 11.8 shows the result of this command.

Showing Lists of Packages

Listing 11.8 Use **yum list packagename** for Information About Installed and Available Versions

```
[root@localhost ~]# yum list kernel
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror.triple-it.nl
 * extras: centos.mirror.triple-it.nl
 * updates: centos.mirror.triple-it.nl
Installed Packages
kernel.x86_64                    3.10.0-123.el7
@anaconda
Available Packages
kernel.x86_64
```

Updating Packages

One of the major benefits of working with yum repositories is that repositories make it easy to update packages. The maintainer of the repositories copies updated packages to the repositories. The index in the repository always contains the current version of a package in the repository. On the local machine also, a database is available with the current versions of the packages that are used. When using the **yum update** command, current versions of packages that are installed are compared to the version of these packages in the repositories. As shown in Listing 11.9, yum next shows an overview of updatable packages. From this overview, type **y** to install the updates.

Notice that while updating packages the old version of the package is replaced with a newer version of the package. There is one exception, which is for the kernel package. Even if you are using the **yum update kernel** command, the kernel package is not updated, but the newer kernel is installed beside the old kernel, so that while booting you can select the kernel that you want to use.

Updating Packages

Listing 11.9 Using **yum update**

```
systemd-sys          x86_64     208-11.el7_0.6      updates       36 k
tuned                noarch     2.3.0-11.el7_0.3    updates      145 k
 tzdata              noarch     2015a-1.el7_0       updates      432 k
 wpa_supplicant      x86_64     1:2.0-13.el7_0      updates      801 k
 yum-plugin-fastestmirror       noarch      1.1.31-25.el7_0
updates       28 k

Transaction Summary
======================================================================
Install    1 Package
Upgrade   79 Packages

Total download size: 95 M
Is this ok [y/d/N]:
```

Working with *yum* Package Groups

While managing specific services on a Linux machine, you often need several different packages. If, for instance, you want to make your machine a virtualization host, you need the KVM packages, but also all supporting packages such as qemu, libvirt, and the client packages. Or while configuring your server as a web server, you need to install additional packages like PHP as well in many cases.

To make it easier to manage specific functionality, instead of specific packages, you can work with package groups as well. A package group is defined in the repository, and yum offers the group management commands to work with these groups. For an overview of all current groups, use **yum groups list**. This shows output as in Listing 11.10.

TIP The name of the command is **yum groups**, but there are aliases that ensure that **yum group** and even commands like **yum groupinstall** are also working. So, you can use either of these commands.

Working with *yum* Package Groups

Listing 11.10 Showing Available yum Groups

```
[root@localhost ~]# yum groups list
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror.triple-it.nl
 * extras: centos.mirror.triple-it.nl
 * updates: centos.mirror.triple-it.nl
Available environment groups:
   Minimal Install
   Infrastructure Server
   File and Print Server
   Basic Web Server
   Virtualization Host
   Server with GUI
   GNOME Desktop
   KDE Plasma Workspaces
   Development and Creative Workstation
Installed groups:
   Console Internet Tools
   System Administration Tools
Available Groups:
   Compatibility Libraries
```

Working with *yum* Package Groups

Notice that some yum groups are not listed by default. To show those as well, type **yum groups list hidden.** You see the list of groups that is displayed is considerably longer. The difference is that **yum groups list** shows environment groups, which contain basic functionality. Within an environment group, different subgroups can be used; these are displayed only when using **yum groups list hidden.**

To get information about packages available in a group, you use **yum groups info.** Because group names normally contain spaces, do not forget to put the entire group name between quotes. So, type **yum groups info "Basic Web Server"** to see what is in the Basic Web Server group. As shown in Listing 11.11, this command shows mandatory items and optional items in the group. The items can be groups and individual packages.

Working with *yum* Package Groups

Listing 11.11 Showing Group Contents with **yum groups info**

```
[root@localhost ~]# yum groups info "Basic Web Server"
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror.triple-it.nl
 * extras: centos.mirror.triple-it.nl
 * updates: centos.mirror.triple-it.nl

Environment Group: Basic Web Server
 Environment-Id: web-server-environment
 Description: Server for serving static and dynamic internet content.
 Mandatory Groups:
   +base
    core
    web-server
 Optional Groups:
   +backup-client
```

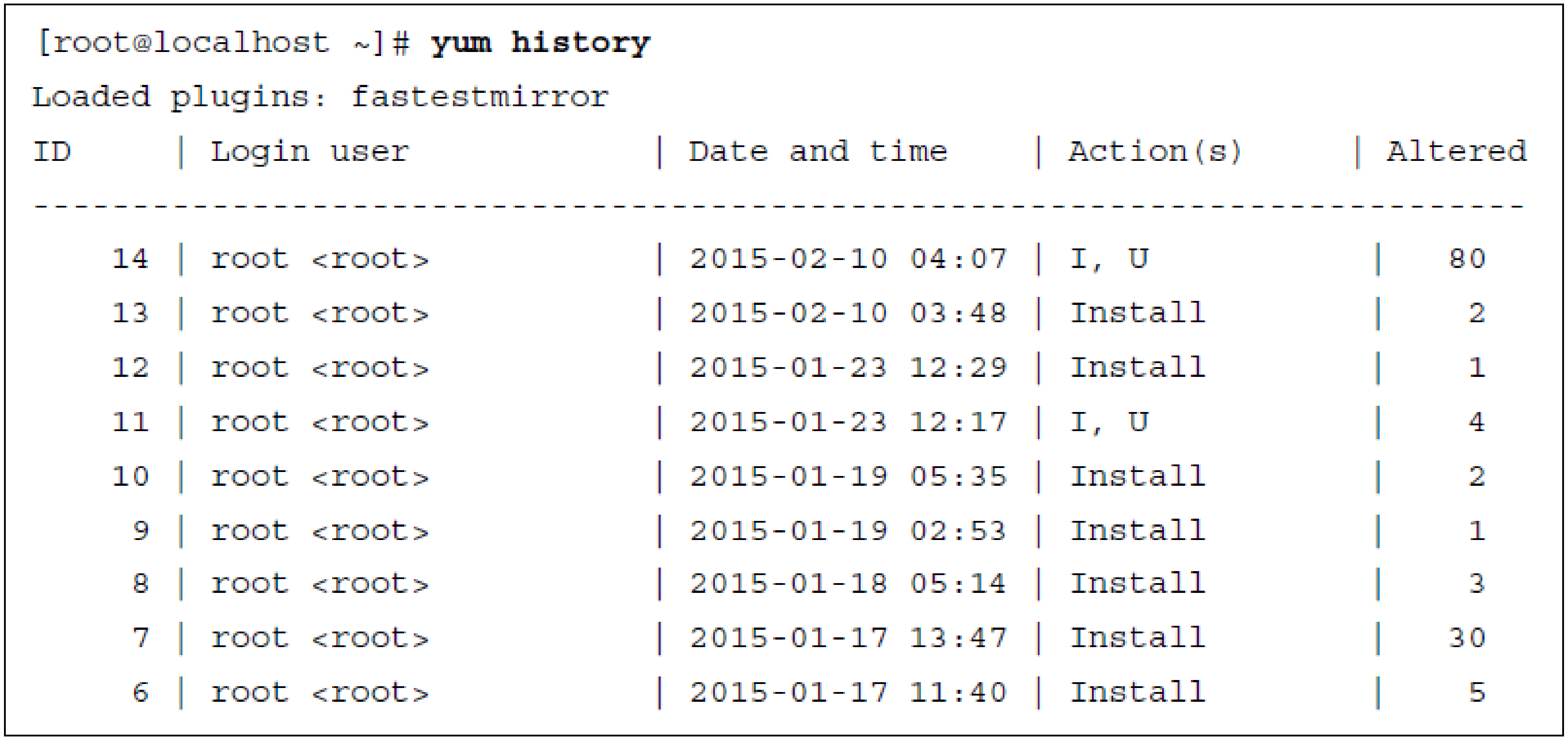
Using *yum* History

While working with yum, all actions are logged to the /var/log/yum.log file. You can use the **yum history** command to get an overview of all actions that have been issued. From the history file, it is possible to undo specific actions; use **yum history undo** followed by the number of the specific action you want to undo.

In Listing 11.12, you see the result of the **yum history** command where every action has its own ID.

Using *yum* History

Listing 11.12 Showing Past yum Actions Using **yum history**



```
[root@localhost ~]# yum history
Loaded plugins: fastestmirror
ID     | Login user               | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
    14 | root <root>              | 2015-02-10 04:07 | I, U           |   80
    13 | root <root>              | 2015-02-10 03:48 | Install        |    2
    12 | root <root>              | 2015-01-23 12:29 | Install        |    1
    11 | root <root>              | 2015-01-23 12:17 | I, U           |    4
    10 | root <root>              | 2015-01-19 05:35 | Install        |    2
     9 | root <root>              | 2015-01-19 02:53 | Install        |    1
     8 | root <root>              | 2015-01-18 05:14 | Install        |    3
     7 | root <root>              | 2015-01-17 13:47 | Install        |   30
     6 | root <root>              | 2015-01-17 11:40 | Install        |    5
```

As you can see, action number 14 altered 80 packages and was used to install and update packages.
To undo this action completely, type **yum history undo 14**

Exercise 11.2 Using yum for Package Management

In this exercise, you use yum for common package management tasks.

1. Type **yum repolist** to show a list of the current repositories that your system is using.
2. Type **yum search xeyes**. This will give no matching result.
3. Type **yum provides */xeyes**. The command shows that the xorg-x11-apps-<version> package contains this file.
4. Install this package using **yum install -y xorg-x11-apps**. Depending on your current configuration, you might notice that quite a few dependencies have to be installed also.
5. Type **yum list xorg-x11-apps**. You see that the package is listed as installed.
6. Type **yum history** and notice the number of the last **yum** command you used.
7. Type **yum history undo <nn>** (where **<nn>** is replaced with the number that you found in step 6). This undoes the last action, so it removes the package you just installed.
8. Repeat the **yum list xorg-x11-apps** command. The package is now listed as available but not as installed.

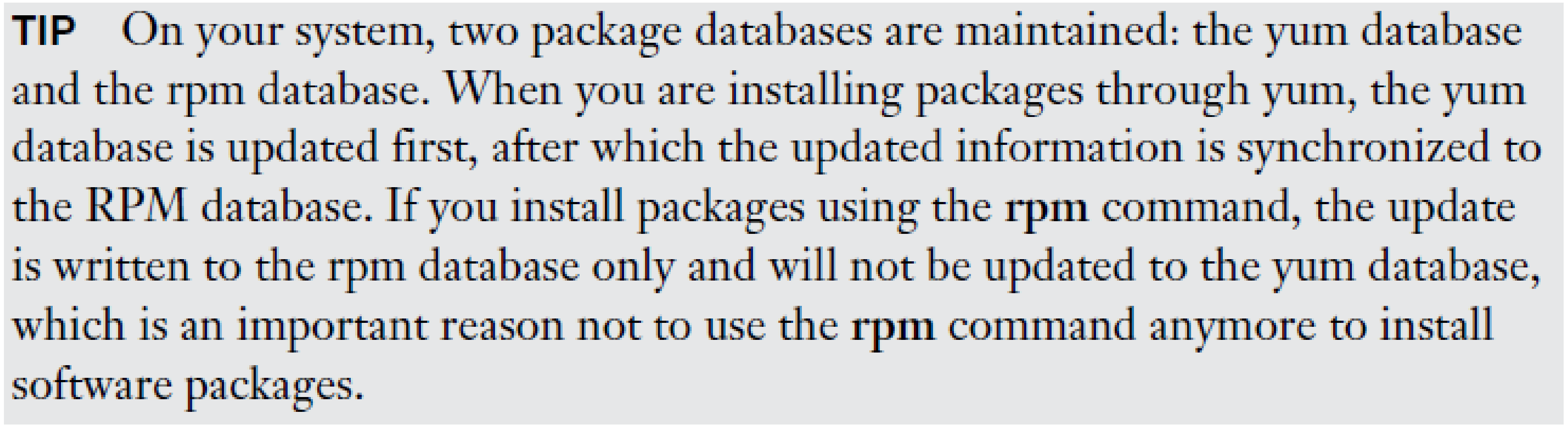
Managing Software Packages with *rpm*

Once upon a time, repositories did not exist, and the **rpm** command was used to install package files after they had been downloaded. That worked, but there was one major issue: the dependency hell.

On modern RHEL systems, repositories are used, and packages are installed using yum. The **yum** command considers all package dependencies and tries to look them up in the currently available repositories. On an RHEL system configured to get updates from the RHN network, or on a CentOS system where consistent repositories are used, the result is that package installation nowadays is without problems and the RPM command is needed no longer for software installation.

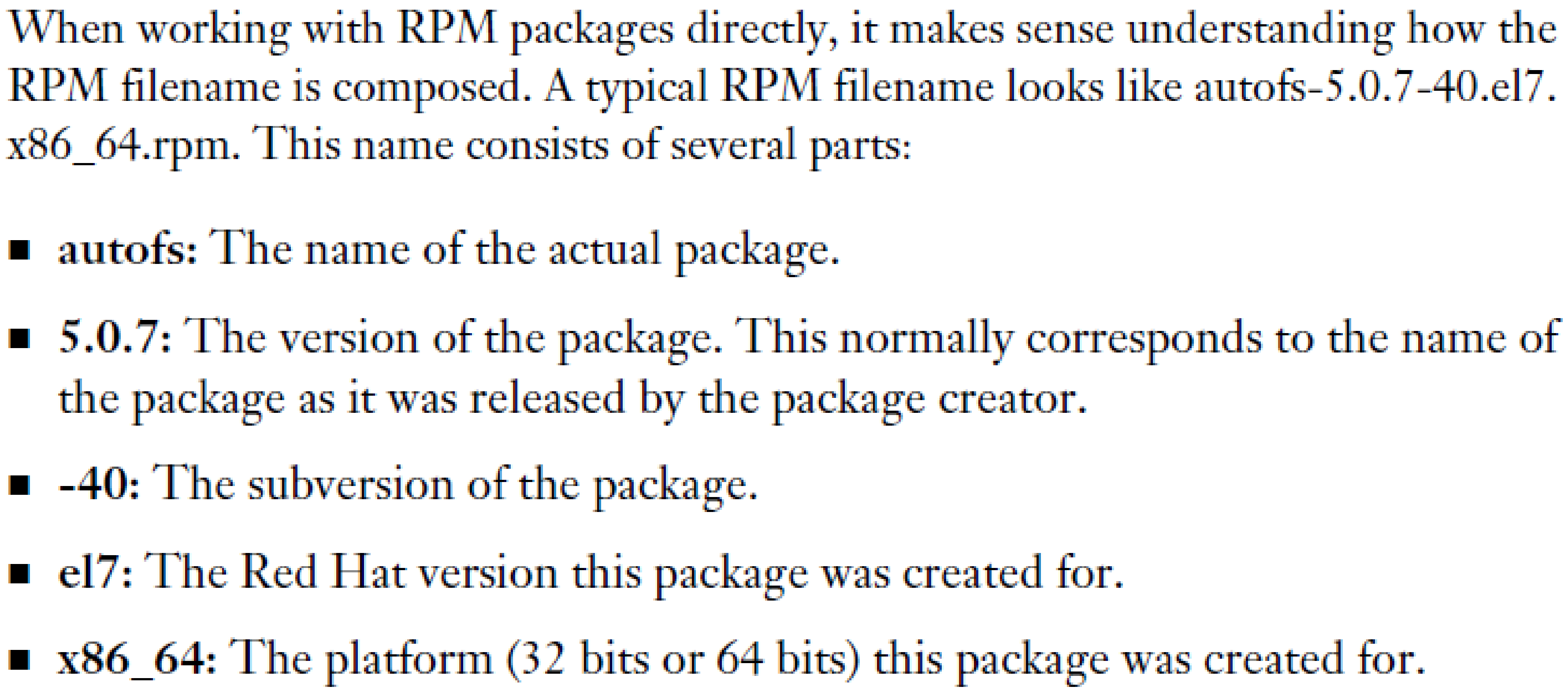
Even after downloading an RPM package file, you do not need to use the **rpm -Uvh packagename** command anymore to install it (even if it still works). A much better alternative is **yum install packagename**, which installs the package and also considers the repositories to resolve dependencies automatically. (In earlier versions of RHEL, the **yum localinstall** command was used to do this; in RHEL 7, **yum localinstall** was deprecated.) That does not mean the **rpm** command has become totally useless. You can still use it to query RPM packages.

Managing Software Packages with *rpm*



TIP On your system, two package databases are maintained: the yum database and the rpm database. When you are installing packages through yum, the yum database is updated first, after which the updated information is synchronized to the RPM database. If you install packages using the **rpm** command, the update is written to the rpm database only and will not be updated to the yum database, which is an important reason not to use the **rpm** command anymore to install software packages.

Understanding RPM Filenames



When working with RPM packages directly, it makes sense understanding how the RPM filename is composed. A typical RPM filename looks like autofs-5.0.7-40.el7.x86_64.rpm. This name consists of several parts:

- **autofs:** The name of the actual package.
- **5.0.7:** The version of the package. This normally corresponds to the name of the package as it was released by the package creator.
- **-40:** The subversion of the package.
- **el7:** The Red Hat version this package was created for.
- **x86_64:** The platform (32 bits or 64 bits) this package was created for.

Querying the RPM Database

The **rpm** command enables you to get much information about packages. Using RPM queries can be a really useful way to find out how software can be configured and used. To start, you can use the **rpm -qa** command. Like **yum list installed**, this shows a list of all software that is installed on the machine. Use **grep** on this command to find out specific package names. To perform queries on RPM packages, you just need the name and not the version information.

After finding the package about which you want to have more information, you can start with some generic queries to find out what is in the package. In the following examples, I assume that you are using RPM queries on the nmap RPM package. To start, type **rpm -qi nmap** to get a description of the package.

The next step is to use **rpm -ql nmap**, which shows a list of all files that are in the package. On some packages, the result can be a really long list of filenames that is not particularly useful. To get more specific information, use **rpm -qd nmap**, which shows all documentation available for the package, or **rpm -qc nmap**, which shows all configuration files in the package.

Querying RPM Package Files

Table 11.5 Common RPM Query Commands

Command	Use
rpm -qf	Uses a filename as its argument to find the specific RPM package a file belongs to.
rpm -ql	Uses the RPM database to provide a list of files in the RPM package.
rpm -qi	Uses the RPM database to provide package information (equivalent to yum info).
rpm -qd	Uses the RPM database to show all documentation that is available in the package.
rpm -qc	Uses the RPM database to show all configuration files that are available in the package.
rpm -q --scripts	Uses the RPM database to show scripts that are used in the package. Particularly useful if combined with the **-p** option.
rpm -qp ...	The **-p** option is used with all the previously listed options to query individual RPM package files instead of the RPM package database. Using this option before installation helps you find out what is actually in the package before it is installed.
rpm -qR	Shows dependencies for a specific package.
rpm -V	Use on an individual package to see which parts of the package have been changed since installation.
rpm -Va	Verifies all installed packages and shows which parts of the package have been changed since installation. This is an easy and convenient way to do a package integrity check.
rpm -qa	Lists all packages that are installed on this server

Using *repoquery*

While **rpm -qp** provides useful tools to query packages before installation, there is a slight problem with this command: It works only on RPM package files, and it cannot query files directly from the repositories. If you want to query packages from the repositories before they have been installed, you need **repoquery**. This binary is not installed by default, so make sure to install the **yum-utils** RPM package to use it.

The **repoquery** command is pretty similar to the **rpm -q** command and uses many similar options. There is just one significant option missing: the **--script** option. A simple solution is to make sure that you are using trusted repositories only, to prevent installing software that contains dangerous script code.

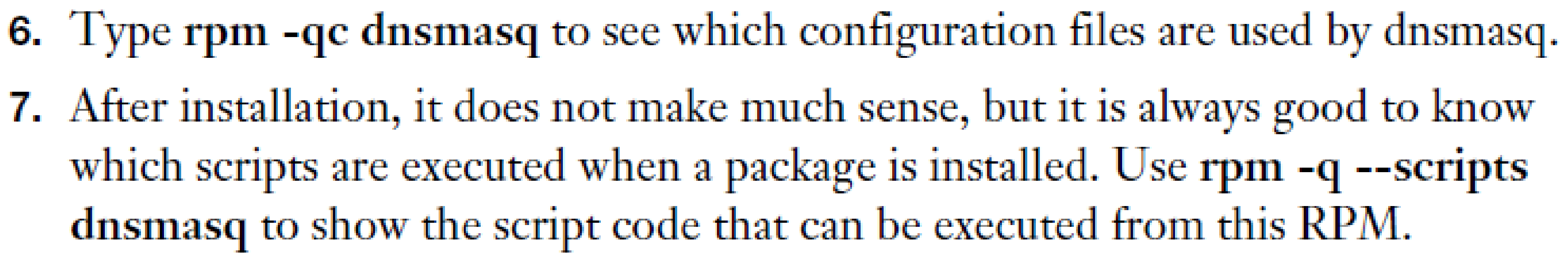
If you need to thoroughly analyze what an RPM package is doing when it is installed, you can download it to your machine, which allows you to use the **rpm -qp --scripts** command on the package. To download a package from the repository to the local directory, you can use the **yumdownloader** command, which comes from the **yum-utils** package.

Exercise 11.3 Using RPM Queries

In this exercise, you learn how to use RPM queries to get more information about software that is installed on your RHEL system.

1. Type **which dnsmasq**. This command gives the complete path name of the **dnsmasq** command.
2. Type **rpm -qf $(which dnsmasq)**. This will do an RPM file query on the result of the **which dnsmasq** command; you learn more about this technique in Chapter 31, "An Introduction to Bash Shell Scripting."
3. Now that you know that the dnsmasq binary comes from the dnsmasq package, use **rpm -qi dnsmasq** to show more information about the package.
4. The information that is shown with **rpm -qi** is useful, but it does not give the details that are needed to start working with the software in the package. Use **rpm -ql dnsmasq** to show a list of all files in the package.
5. Use **rpm -qd dnsmasq** to show the available documentation. Notice that this command reveals that there is a man page, but there is also a doc.html and a setup.html file in the /usr/share/doc/dnsmasq-version directory. Open these files with your browser to get more information about the use of dnsmasq.

Exercise 11.3 Using RPM Queries



6. Type **rpm -qc dnsmasq** to see which configuration files are used by dnsmasq.
7. After installation, it does not make much sense, but it is always good to know which scripts are executed when a package is installed. Use **rpm -q --scripts dnsmasq** to show the script code that can be executed from this RPM.

Summary

In this chapter, you learned how to work with software on Red Hat Enterprise Linux. You learned how to use yum to manage software packages coming from repositories. You also learned how to use the **rpm** command to perform queries on the packages on your system. Make sure that you master these essential skills well; they are key to getting things done on Red Hat Enterprise Linux.

Define Key Terms

- Define the following key terms:
  - yum,
  - repository,
  - dependency,
  - package,
  - Red Hat Network,
  - package groups,
  - dependency hell,
  - RPM

Lab 11.1

1. Copy some RPM files from the installation disk to the /myrepo directory. Make this directory a repository and make sure that your server is using this repository.
2. List the repositories currently in use on your server.
3. Search for the package that contains the cache only DNS name server. Do not install it yet.
4. Perform an extensive query of the package so that you know before you install it which files it contains, which dependencies it has, and where to find the documentation and configuration.
5. Check whether the RPM contains any scripts. You may download it, but you may not install it yet; you want to know which scripts are in a package before actually installing it, right?
6. Install the package you have found in step 3.
7. Undo the installation.

Chapter 12:

Scheduling Tasks

Chapter 12 Objectives

- The following topics are covered in this chapter:
  - Scheduling once-only future tasks with at
  - Scheduling regular future tasks with cron
- The following RHCSA exam objectives are covered in this chapter:
  - Scheduling tasks using at and cron

Scheduling Tasks

On a Linux server it is important that certain tasks run at certain times. This can be done by using the atd and crond services, which can be configured to run tasks in the future. The atd service is for executing future tasks once only, the crond service is for recurring regular tasks. In this chapter you learn how to configure both.

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 two major components. First, there is the cron daemon crond. This daemon 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. In this section, you learn how to configure cron.

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, a service that cleans up log files and runs on a regular basis, but other important maintenance processes are started automatically through cron also.

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.

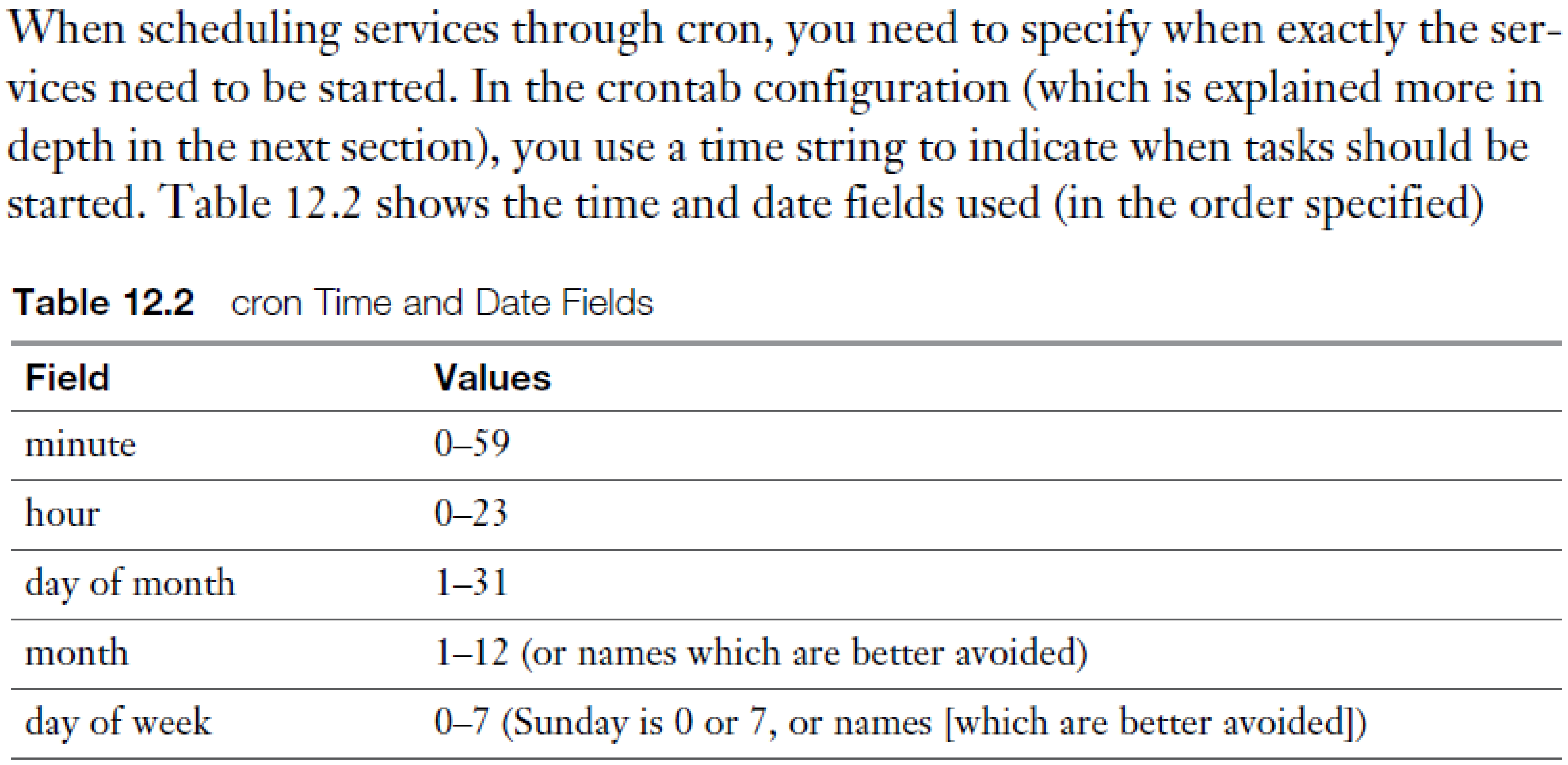
To monitor the current status of the cron service, you can use the **systemctl status crond -l** command. Listing 12.1 shows the output of this command.

Managing the *cron* Service

Listing 12.1 Monitoring the Current State of the crond Service

```
[root@server2 ~]# systemctl status crond -l
crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled)
   Active: active (running) since Wed 2015-02-11 03:50:14 EST; 5 days
ago
 Main PID: 550 (crond)
   CGroup: /system.slice/crond.service
           └─550 /usr/sbin/crond -n
```

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 12.2 shows the time and date fields used (in the order specified)

Table 12.2 cron Time and Date Fields

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])

Understanding *cron* Timing

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 (probably not what you want)
- **0 11 * * 1-5** Every day at 11 a.m. on weekdays only
- **0 7-18 * * 1-5** Every hour on weekdays on the hour
- **0 */2 2 12 5** Every 2 hours on the hour on December second and every Friday in December

TIP No need trying to remember all this, **man 5 crontab** shows all possible constructions.

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 (see Listing 12.2). To make modifications to the cron jobs, there are other locations where cron jobs should be specified.

Managing *cron* Configuration Files

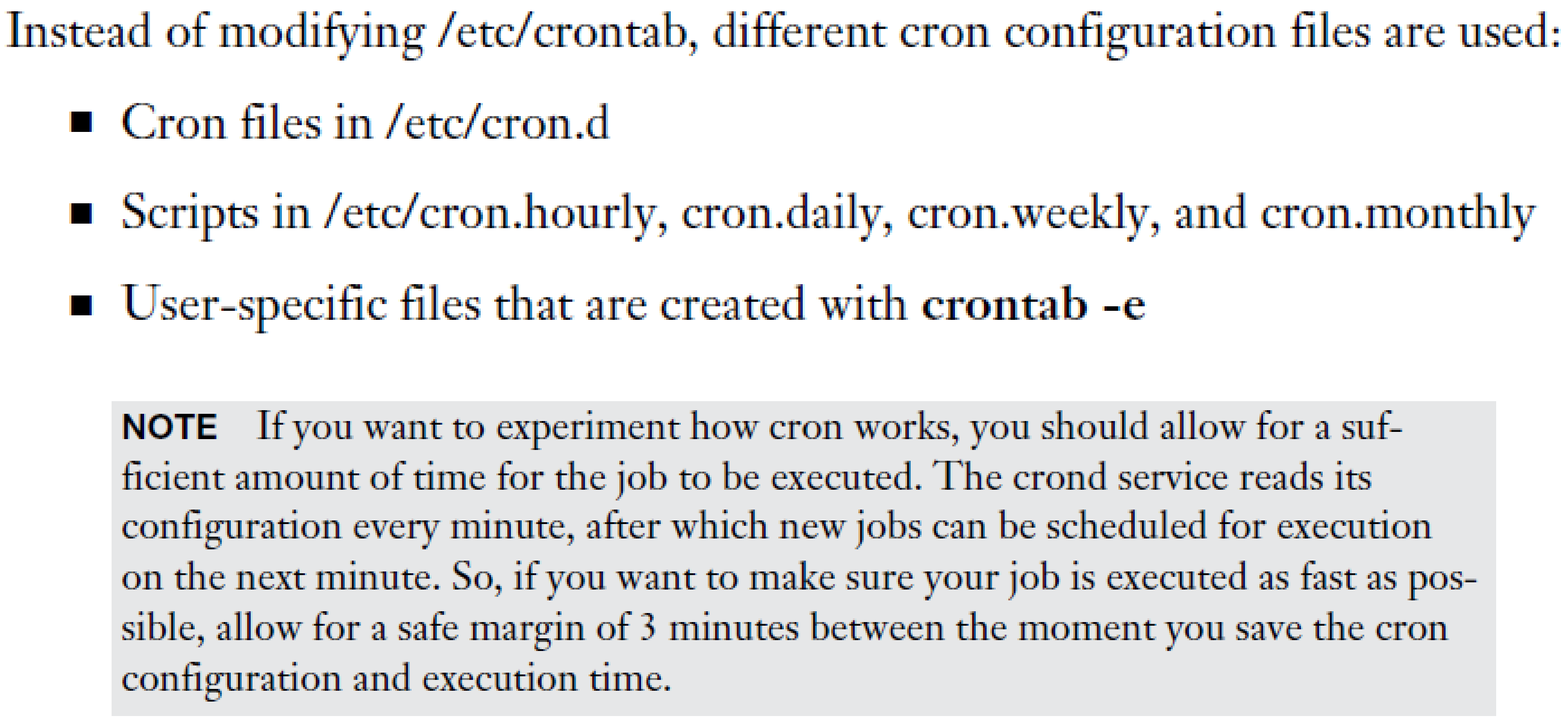
Listing 12.2 /etc/crontab Sample Content

```
[root@server2 ~]# 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
```

Managing *cron* Configuration Files



Instead of modifying /etc/crontab, different cron configuration files are used:

- 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**

NOTE If you want to experiment how cron works, you should allow for a sufficient amount of time for the job to be executed. The crond service reads its configuration every minute, after which new jobs can be scheduled for execution on the next minute. So, if you want to make sure your job is executed as fast as possible, allow for a safe margin of 3 minutes between the moment you save the cron configuration and execution time.

Managing *cron* Configuration Files

To start, cron jobs can be started for specific users. 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.

Managing *cron* Configuration Files

Listing 12.3 Example cron Job in /etc/cron.d

```
[root@server1 cron.d]# cat 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. First there is the time indication, which has the command start at 3:10 a.m. on the first of every month. Then, the configuration indicates that the command has to be started as the unbound user. 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.

Managing *cron* Configuration Files

The last way to schedule cron jobs is through the following directories:

- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /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.

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. Listing 12.4 shows the contents of the /etc/anacrontab file, which is used to specify how anacrontab jobs should be executed.

Understanding the Purpose of *anacron*

Listing 12.4 anacrontab Configuration

```
[root@server1 spool]# 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
```

Exercise 12.1 Running Scheduled Tasks Through *cron*

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)
```

Exercise 12.1 Running Scheduled Tasks Through *cron*

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/crond.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.

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 RHEL 7, 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. In Exercise 12.2, you learn how to work with at to schedule jobs for execution at a specific time.

Exercise 12.2 Scheduling Jobs with at

1. Type **systemctl status atd**. 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.

Summary

In this chapter, you learned how to schedule jobs for future execution. You also learned how to configure cron to execute jobs repeatedly at a specific time. You learned that different methods exist to tell cron when a job should be executed. In addition, you learned about the anacron service, which is used to make sure that the jobs in the directories /etc/cron.{hourly|daily|weekly|monthly} are indeed executed, even if the system temporarily is not available. At the end of this chapter, you learned how to use the **atd** service to schedule tasks to be executed once.

Define Key Terms

- Define the following key terms:
  - cron,
  - anacron,
  - at

Lab 12.1

1. Create a cron job that performs an update of all software on your computer every evening at 11 p.m.
2. Schedule your machine to be rebooted at 3 a.m. tonight.

Chapter 13:

Configuring Logging

Chapter 13 Objectives

- The following topics are covered in this chapter:
  - Understand System Logging
  - Configuring rsyslogd
  - Rotating Log Files
  - Working with journald

- The following RHCSA exam objectives are covered in this chapter:
  - Locate and interpret system log files and journals

Configuring Logging

Analyzing log files is an important system administrator task. If anything goes wrong on a Linux system, the answer is often in the log files. On RHEL 7, two different log systems are used side by side, and it is important to know which information can be found where. This chapter teaches you all about it. You learn how to read log files, configure rsyslogd and journald, and also how to set up your system for log rotation so that you can prevent your disks from being completely filled up by services that are logging too enthusiastically.

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:** rsyslogd is the enhancement of syslogd, a service that takes care of managing centralized log files. Syslogd has been around for a long time.
- **journald:** With the introduction of systemd, the journald log service systemd-journald has been introduced also. 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.

Understanding the Role of *rsyslogd* and *journald*

On RHEL 7, journald (which is implemented by the systemd-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. This event journal is stored in a binary format, and it can be queried using the **journalctl** command.

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 Red Hat Enterprise Linux 7, 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.

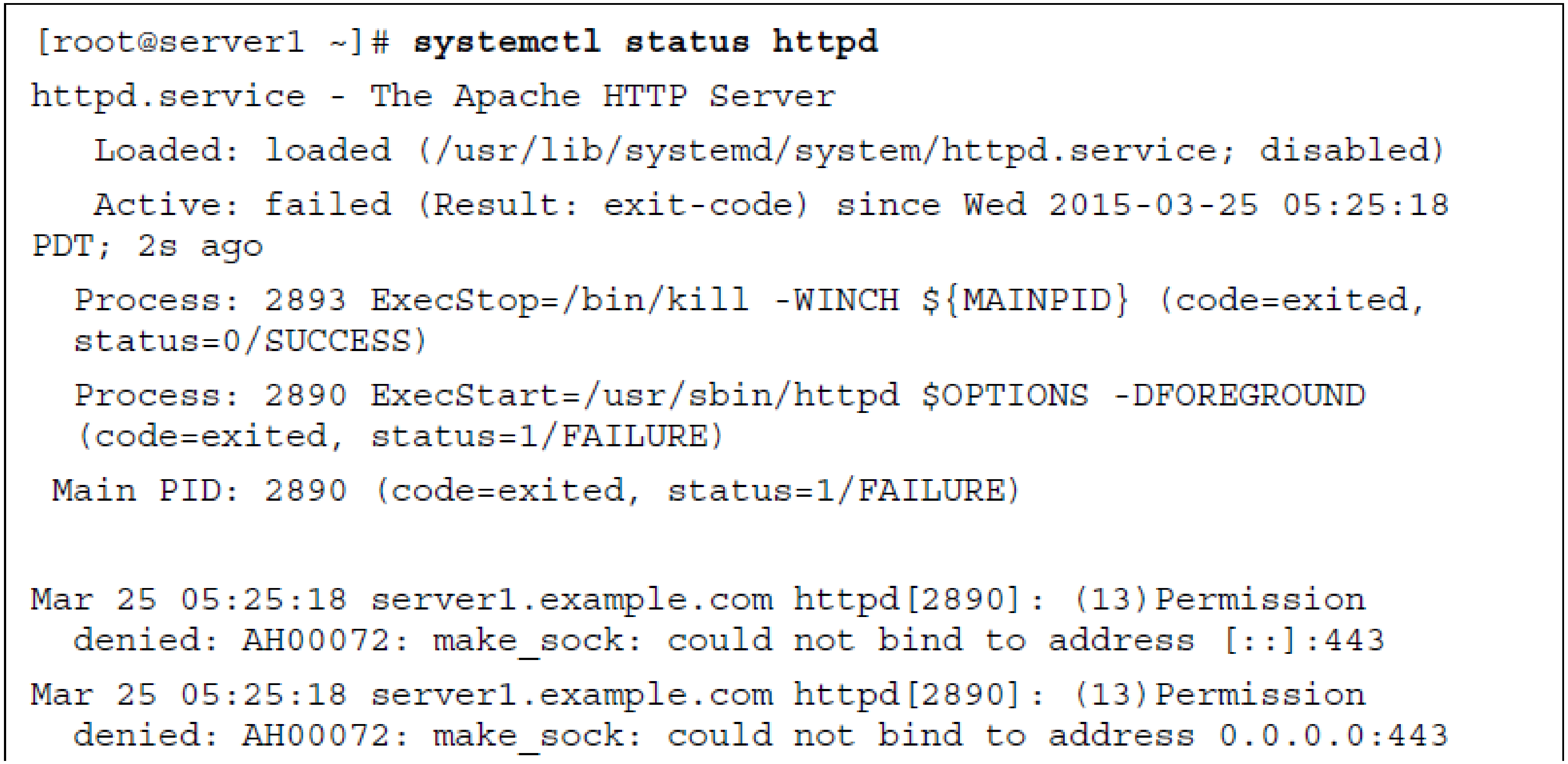
Understanding the Role of *rsyslogd* and *journald*

To get more information about what has been happening on a machine running RHEL, administrators of Red Hat Enterprise Linux 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. Listing 13.1 shows an example where this command clearly indicates what went wrong while starting a service.

Understanding the Role of *rsyslogd* and *journald*

Listing 13.1 Using **systemctl** Status to Show Relevant Log Information



```
[root@server1 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: failed (Result: exit-code) since Wed 2015-03-25 05:25:18
PDT; 2s ago
  Process: 2893 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited,
  status=0/SUCCESS)
  Process: 2890 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
  (code=exited, status=1/FAILURE)
 Main PID: 2890 (code=exited, status=1/FAILURE)


Mar 25 05:25:18 server1.example.com httpd[2890]: (13)Permission
  denied: AH00072: make_sock: could not bind to address [::]:443
Mar 25 05:25:18 server1.example.com httpd[2890]: (13)Permission
  denied: AH00072: make_sock: could not bind to address 0.0.0.0:443
```

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**.

Reading Log Files

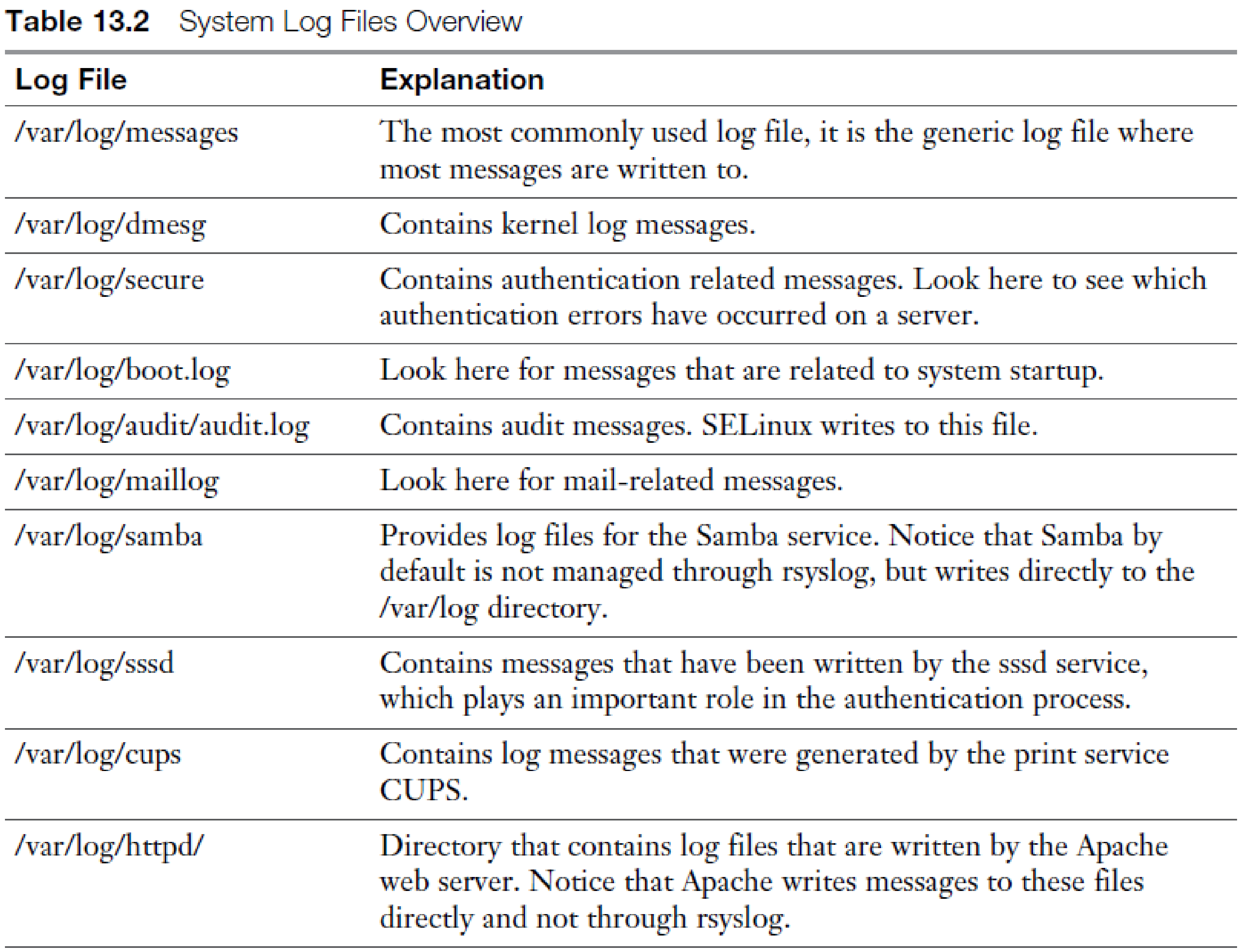
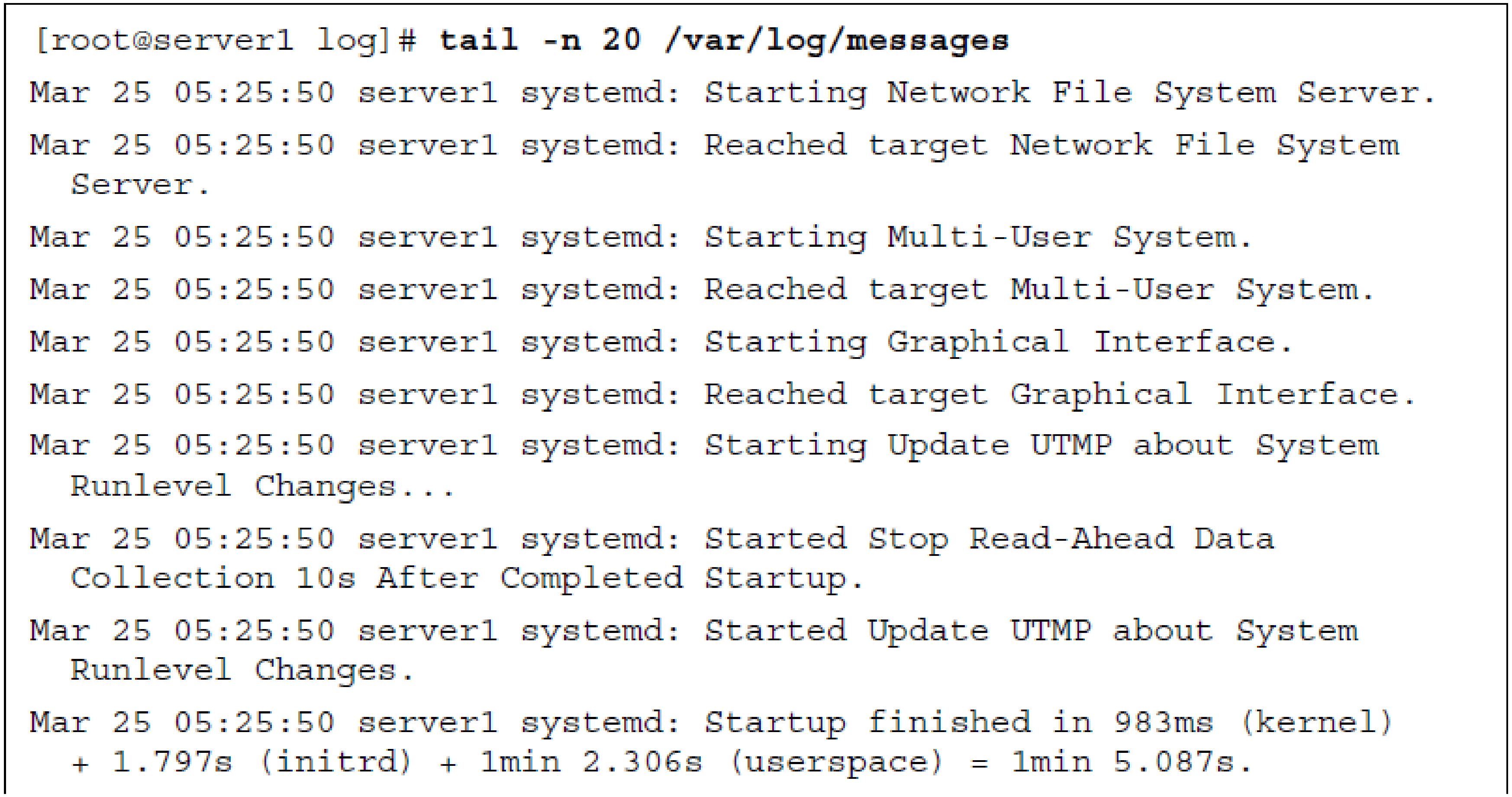


Table 13.2 System Log Files Overview

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.

Understanding Log File Contents

Listing 13.2 /var/log/messages Sample Content



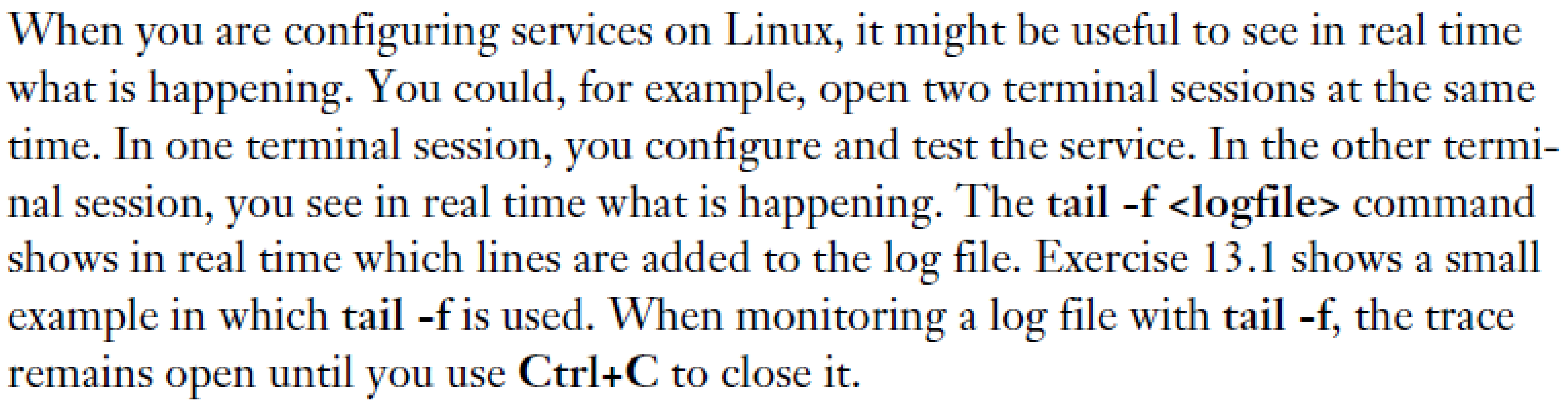
```
[root@server1 log]# tail -n 20 /var/log/messages
Mar 25 05:25:50 server1 systemd: Starting Network File System Server.
Mar 25 05:25:50 server1 systemd: Reached target Network File System
  Server.
Mar 25 05:25:50 server1 systemd: Starting Multi-User System.
Mar 25 05:25:50 server1 systemd: Reached target Multi-User System.
Mar 25 05:25:50 server1 systemd: Starting Graphical Interface.
Mar 25 05:25:50 server1 systemd: Reached target Graphical Interface.
Mar 25 05:25:50 server1 systemd: Starting Update UTMP about System
  Runlevel Changes...
Mar 25 05:25:50 server1 systemd: Started Stop Read-Ahead Data
  Collection 10s After Completed Startup.
Mar 25 05:25:50 server1 systemd: Started Update UTMP about System
  Runlevel Changes.
Mar 25 05:25:50 server1 systemd: Startup finished in 983ms (kernel)
  + 1.797s (initrd) + 1min 2.306s (userspace) = 1min 5.087s.
```

Understanding Log File Contents

As you can see in Listing 13.2, 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.

Live Log File Monitoring

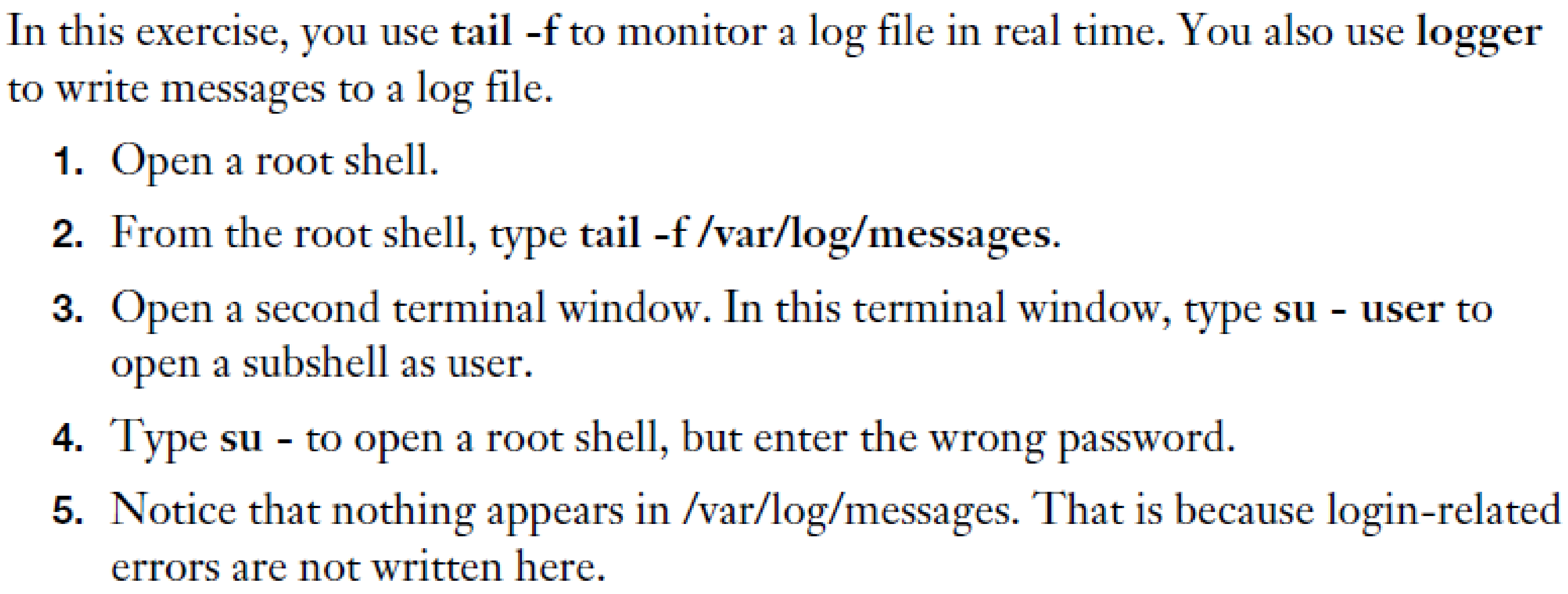


When you are configuring services on Linux, it might be useful to see in real time what is happening. You could, for example, open two terminal sessions at the same time. In one terminal session, you configure and test the service. In the other terminal session, you see in real time what is happening. The **tail -f <logfile>** command shows in real time which lines are added to the log file. Exercise 13.1 shows a small example in which **tail -f** is used. When monitoring a log file with **tail -f**, the trace remains open until you use **Ctrl+C** to close it.

Using *logger*

Most services write information to the log files all by themselves. The **logger** command enables users to write messages to rsyslog from the command line. Using this command is simple. Just type **logger**, followed by the message you want to write to the logs. The logger utility, in this way, offers a convenient solution to write messages from scripts. This allows you to have a script write to syslog if something goes wrong.

Exercise 13.1 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.

Exercise 13.1 Using Live Log Monitoring and logger

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.

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.

Understanding *rsyslogd* Configuration Files

Like many other services on RHEL 7, 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.

Understanding Facilities, Priorities, and Log Destinations

To specify what information should be logged to which destination, rsyslogd uses facilities, priorities, and destinations:

- A *facility* specifies a category of information that is logged. Rsyslogd uses a fixed list of facilities, which cannot be extended. This is because of backward compatibility with the legacy syslog service.
- A *priority* is used to define the severity of the message that needs to be logged. When specifying a priority, by default all messages with that priority and all higher priorities are logged.
- A *destination* defines where the message should be written to. Typical destinations are files, but rsyslog modules can be used as a destination as well, to allow further processing through an rsyslogd module.

Understanding Facilities, Priorities, and Log Destinations

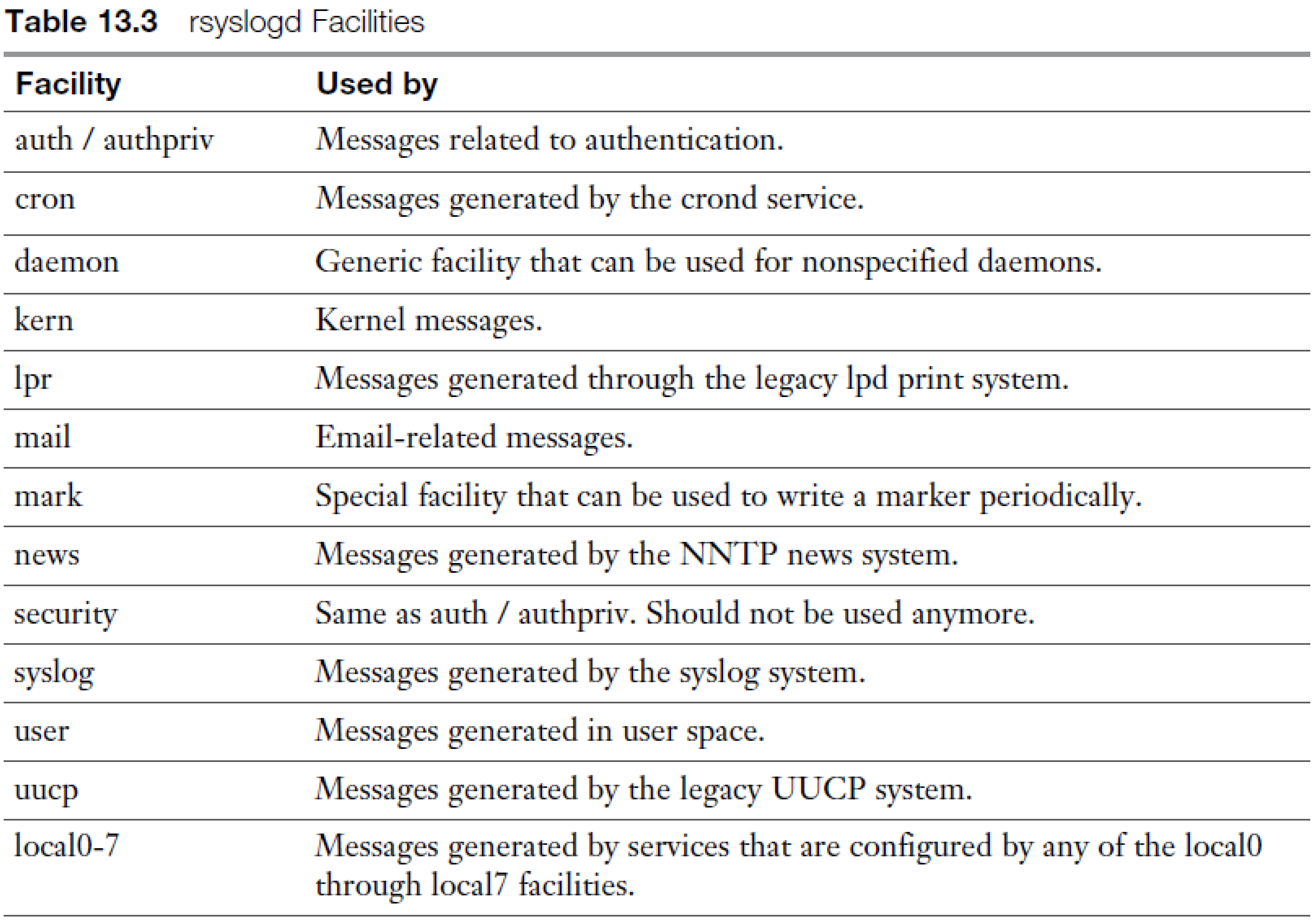


Table 13.3 rsyslogd Facilities

Facility	Used by
auth / authpriv	Messages related to authentication.
cron	Messages generated by the crond service.
daemon	Generic facility that can be used for nonspecified daemons.
kern	Kernel messages.
lpr	Messages generated through the legacy lpd print system.
mail	Email-related messages.
mark	Special facility that can be used to write a marker periodically.
news	Messages generated by the NNTP news system.
security	Same as auth / authpriv. Should not be used anymore.
syslog	Messages generated by the syslog system.
user	Messages generated in user space.
uucp	Messages generated by the legacy UUCP system.
local0-7	Messages generated by services that are configured by any of the local0 through local7 facilities.

Understanding Facilities, Priorities, and Log Destinations

Table 13.4 rsyslogd Priorities

Priority	Used for
debug	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.

TIP There is no need to learn the names of rsyslogd facilities and priorities by heart. They are all listed in man 5 rsyslog.conf. On the exam, you have access to the man pages, so this information will be easily accessible.

Exercise 13.2 Changing rsyslog.conf Rules

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/http/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.**

Exercise 13.2 Changing rsyslog.conf Rules

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/messages-debug" > /etc/rsyslogd/debug.conf**.
9. Again, restart rsyslogd using **systemctl restart rsyslogd**.
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.

Rotating Log Files

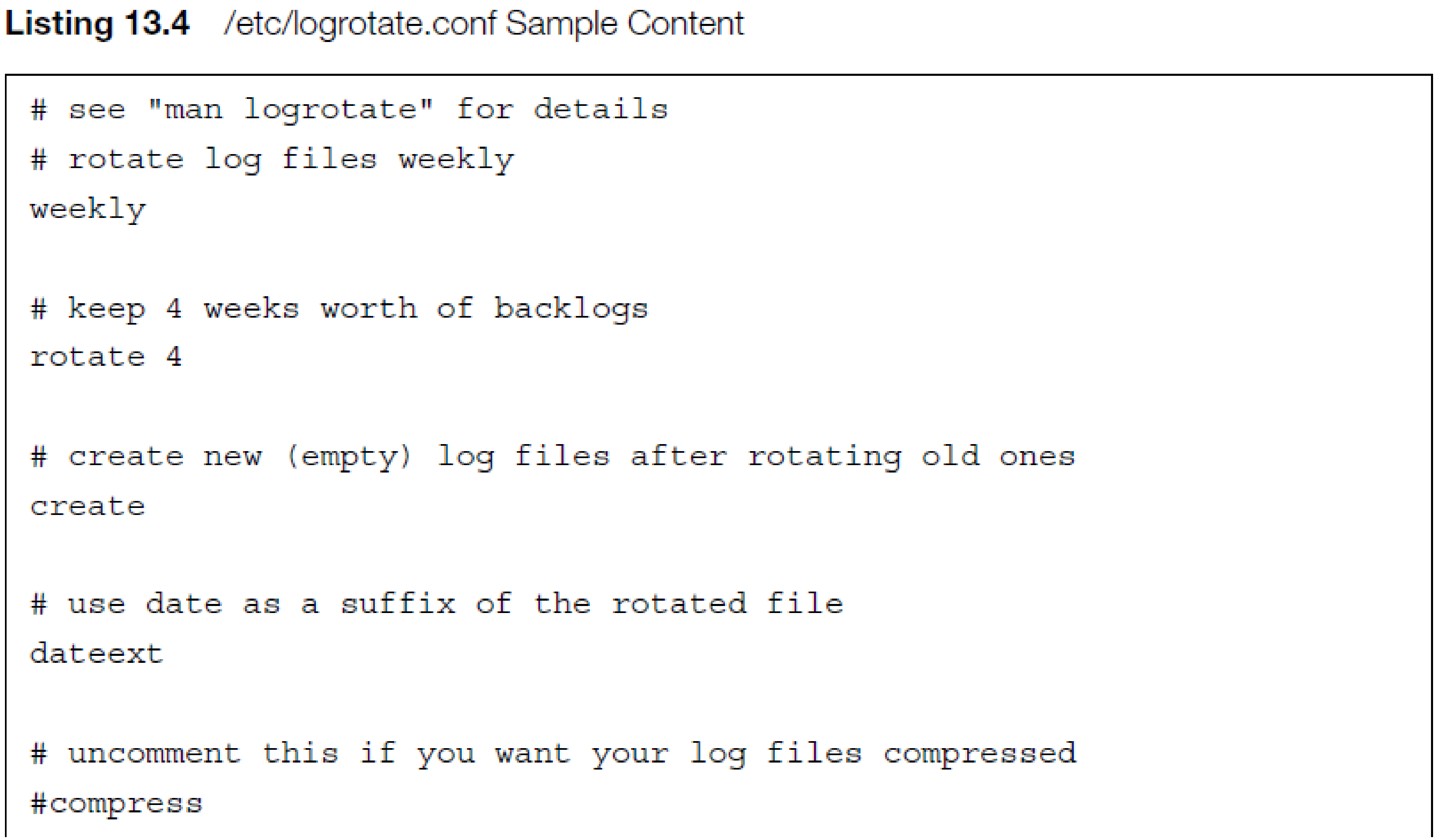
To prevent syslog messages from filling up your system completely, the log messages can be rotated. That means that when a certain threshold has been reached, the old log file is closed and a new log file is opened. The logrotate utility is started periodically through the crond service to take care 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, 2015, the rotated filename will be /var/log/messages-20150115. As a default, four old log files are kept on the system. Files older than that period are removed from the system automatically.

The default settings for log rotation are kept in the file /etc/logrotate.conf (see Listing 13.4).

man logrotate

Rotating Log Files



Listing 13.4 /etc/logrotate.conf Sample Content

```
# 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
```

Rotating Log Files

```
# 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
}
```

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.

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 13.5 shows a partial result of this command.

Using *journalctl* to Find Events

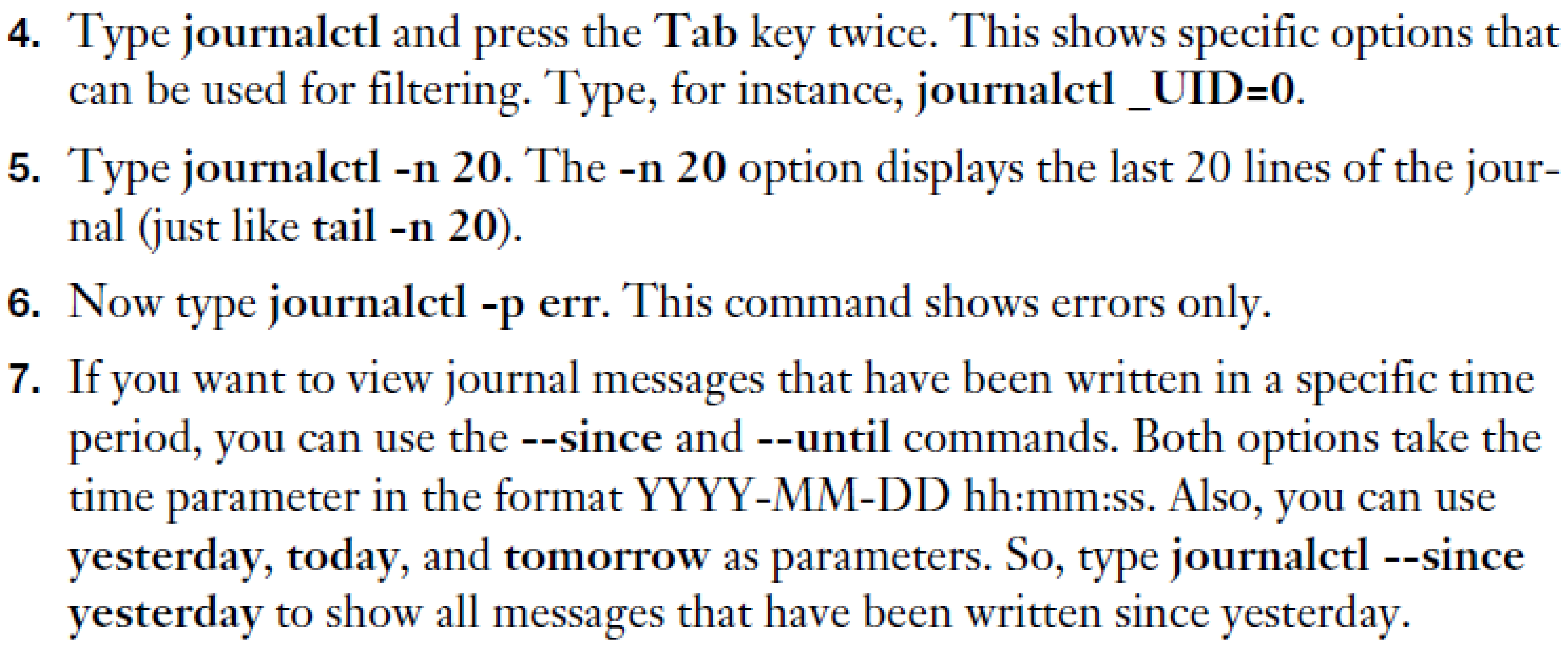
Listing 13.5 Watching Log Information Generated by journald

```
-- Logs begin at Wed 2015-03-25 05:24:43 PDT, end at Wed 2015-03-25
05:46:46 PDT. --
Mar 25 05:24:43 localhost.localdomain systemd-journal[207]: Runtime
journal is using 6.1M (max 49.3M, leaving 74.0M of free 487.3M, cu
Mar 25 05:24:43 localhost.localdomain systemd-journal[207]: Runtime
journal is using 6.1M (max 49.3M, leaving 74.0M of free 487.3M, cu
Mar 25 05:24:43 localhost.localdomain kernel: Initializing cgroup
subsys cpuset
Mar 25 05:24:43 localhost.localdomain kernel: Initializing cgroup
subsys cpu
Mar 25 05:24:43 localhost.localdomain kernel: Initializing cgroup
subsys cpuacct
Mar 25 05:24:43 localhost.localdomain kernel: Linux version 3.10.0-123.
el7.x86_64 (builder@kbuilder.dev.centos.org) (gcc version 4.8.2
Mar 25 05:24:43 localhost.localdomain kernel: Command line: BOOT_
IMAGE=/vmlinuz-3.10.0-123.el7.x86_64 root=UUID=432d640e-3339-45fa-a66
Mar 25 05:24:43 localhost.localdomain kernel: Disabled fast string
operations
Mar 25 05:24:43 localhost.localdomain kernel: e820: BIOS-provided
```

Exercise 13.3 Discovering journalctl

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.

Exercise 13.3 Discovering journalctl



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.

Exercise 13.3 Discovering journalctl

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 (see Listing 13.3). All these 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.

Preserving the systemd Journal

By default, the journal is stored in the file /run/log/journal. The entire /run directory is used for current process status information only, which means that the journal is cleared when the system reboots. To make the journal persistent between system restarts, you should make sure that a directory /var/log/journal exists.

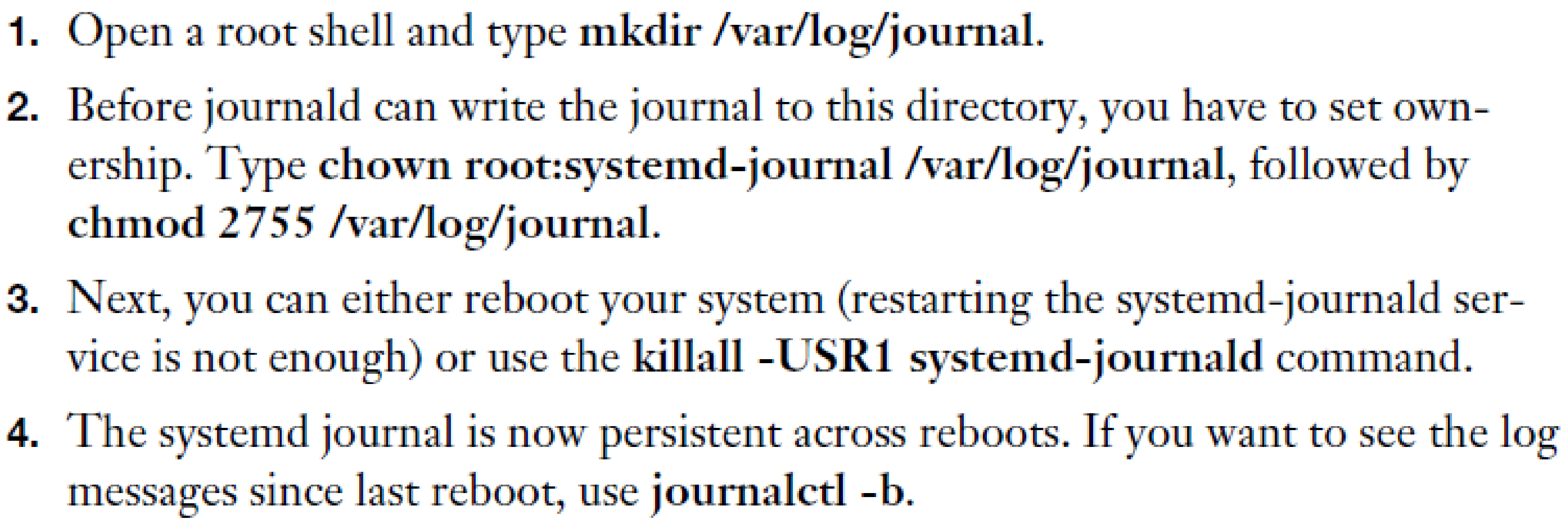
Even when the journal is written to the permanent file in/var/log/journal, that does not mean that the journal is kept forever. The journal has built-in log rotation that will be used monthly. Also, the journal is limited to a maximum size of 10% of the file system size that it is on, and it will also stop growing if less than 15% of the file system is still free. If that happens, the oldest messages from the journal are dropped automatically to make place for newer messages. To change these settings, you can modify the file /etc/systemd/journald.conf. You'll see that in this file some other parameters can be set also (see Listing 13.5).

Preserving the systemd Journal

Listing 13.5 Setting journald Parameters Through /etc/systemd/journald.conf

```
[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=login
#SyncIntervalSec=5m
#RateLimitInterval=30s
#RateLimitBurst=1000
#SystemMaxUse=
#SystemKeepFree=
#SystemMaxFileSize=
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#MaxRetentionSec=
#MaxFileSec=1month
#ForwardToSyslog=yes
#ForwardToKMsg=no
#ForwardToConsole=no
#TTYPath=/dev/console
#MaxLevelStore=debug
#MaxLevelSyslog=debug
#MaxLevelKMsg=notice
#MaxLevelConsole=info
```

Exercise 13.4 Making the journald Journal Permanent



1. Open a root shell and type **mkdir /var/log/journal**.
2. Before journald can write the journal to this directory, you have to set ownership. Type **chown root:systemd-journal /var/log/journal**, followed by **chmod 2755 /var/log/journal**.
3. Next, you can either reboot your system (restarting the systemd-journald service is not enough) or use the **killall -USR1 systemd-journald** command.
4. The systemd journal is now persistent across reboots. If you want to see the log messages since last reboot, use **journalctl -b**.

Summary

In this chapter, you learned how to configure logging. You read how the rsyslogd and journald services are used on RHEL 7 to keep log information, and you learned how to manage logs that are written by these services. You also learned how to configure log rotation and make the journal persistent.

Define Key Terms

- Define the following key terms:
  - journald,
  - journalctl,
  - rsyslogd,
  - facility,
  - priority,
  - destination,
  - log rotation

Lab 13.1

1. Configure the journal to be persistent across system reboots.
2. Make a configuration file that writes all messages with an info priority to the file /var/log/messages.info.
3. Configure logrotate to keep 10 old versions of log files.

Chapter 14:

Managing Partitions

Chapter 14 Objectives

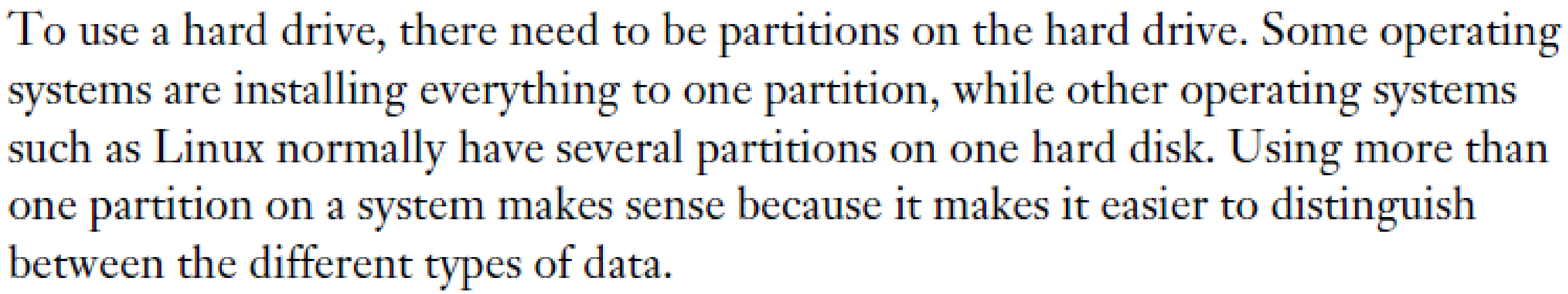
- The following topics are covered in this chapter:
  - Understanding MBR and GPT Partitions
  - Managing Partitions and File Systems
  - Mounting File Systems

- The following RHCSA exam objectives are covered in this chapter:
  - List, create, delete partitions on MBR and GPT disks
  - Configure systems to mount file systems at boot by universal unique ID (UUID) or label
  - Add new partitions and logical volumes, and swap to a system non-destructively
  - Create, mount, unmount, and use vfat, Ext4, and xfs file systems

Managing Partitions

Working with storage is an important task for a Linux administrator. In this chapter, you acquire the first set of essential storage skills. You learn how to create and manage partitions, format them with the file system you need to use, and mount these file systems.

Understanding MBR and GPT Partitions



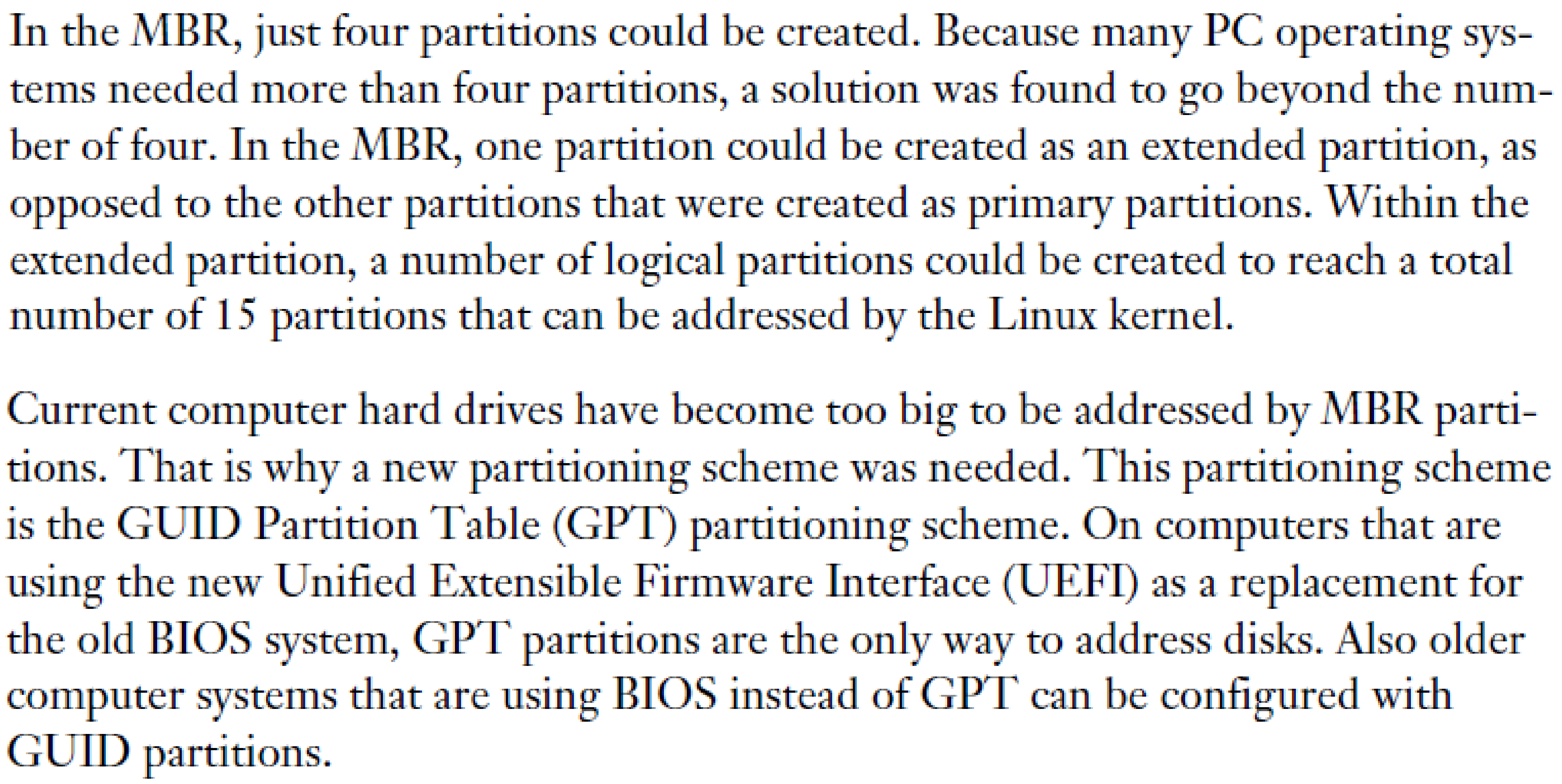
To use a hard drive, there need to be partitions on the hard drive. Some operating systems are installing everything to one partition, while other operating systems such as Linux normally have several partitions on one hard disk. Using more than one partition on a system makes sense because it makes it easier to distinguish between the different types of data.

Understanding the MBR Partitioning Scheme

When the personal computer was invented in early 1982, a system was needed to define hard disk layout. This system became known as the Master Boot Record partitioning scheme. While booting a computer the Basic Input Output System (BIOS) was loaded to access hardware devices. From the BIOS, the bootable device was read, and on this bootable device, the Master Boot Record (MBR) was allocated. The MBR contains all that is needed to start a computer, including a boot loader and a partition table.

The MBR was defined as the first 512 bytes on a computer hard drive, and in the MBR an operating system boot loader (such as GRUB 2; see Chapter 18, "Managing and Understanding the Boot Procedure") was present, as well as a partition table. The size that was used for the partition table was relatively small, just 64 bytes, with the result that in the MBR no more than four partitions could be created. Since partition size data was stored in 32-bit values, and a default sector size of 512 bytes is used, the maximum size that could be used by a partition was limited to 2 TiB (hardly a problem in the early 1980s).

Understanding the Need for GPT Partitioning



In the MBR, just four partitions could be created. Because many PC operating systems needed more than four partitions, a solution was found to go beyond the number of four. In the MBR, one partition could be created as an extended partition, as opposed to the other partitions that were created as primary partitions. Within the extended partition, a number of logical partitions could be created to reach a total number of 15 partitions that can be addressed by the Linux kernel.

Current computer hard drives have become too big to be addressed by MBR partitions. That is why a new partitioning scheme was needed. This partitioning scheme is the GUID Partition Table (GPT) partitioning scheme. On computers that are using the new Unified Extensible Firmware Interface (UEFI) as a replacement for the old BIOS system, GPT partitions are the only way to address disks. Also older computer systems that are using BIOS instead of GPT can be configured with GUID partitions.

Understanding the Need for GPT Partitioning

Using GUID offers many benefits:

- The maximum partition size is 8 zebibyte (ZiB), which is 1024 * 1024 * 1024 * 1024 gibibytes.
- In GPT, up to a maximum number of 128 partitions can be created.
- The 2 TiB limit no longer exists.
- Because space that is available to store partitions is much bigger than 64 bytes, which was used in MBR, there is no longer a need to distinguish between primary, extended, and logical partitions.
- GPT uses a 128-bit global unique ID (GUID) to identify partitions.
- A backup copy of the GUID partition table is created by default at the end of the disk, which eliminates the single point of failure that exists on MBR partition tables.

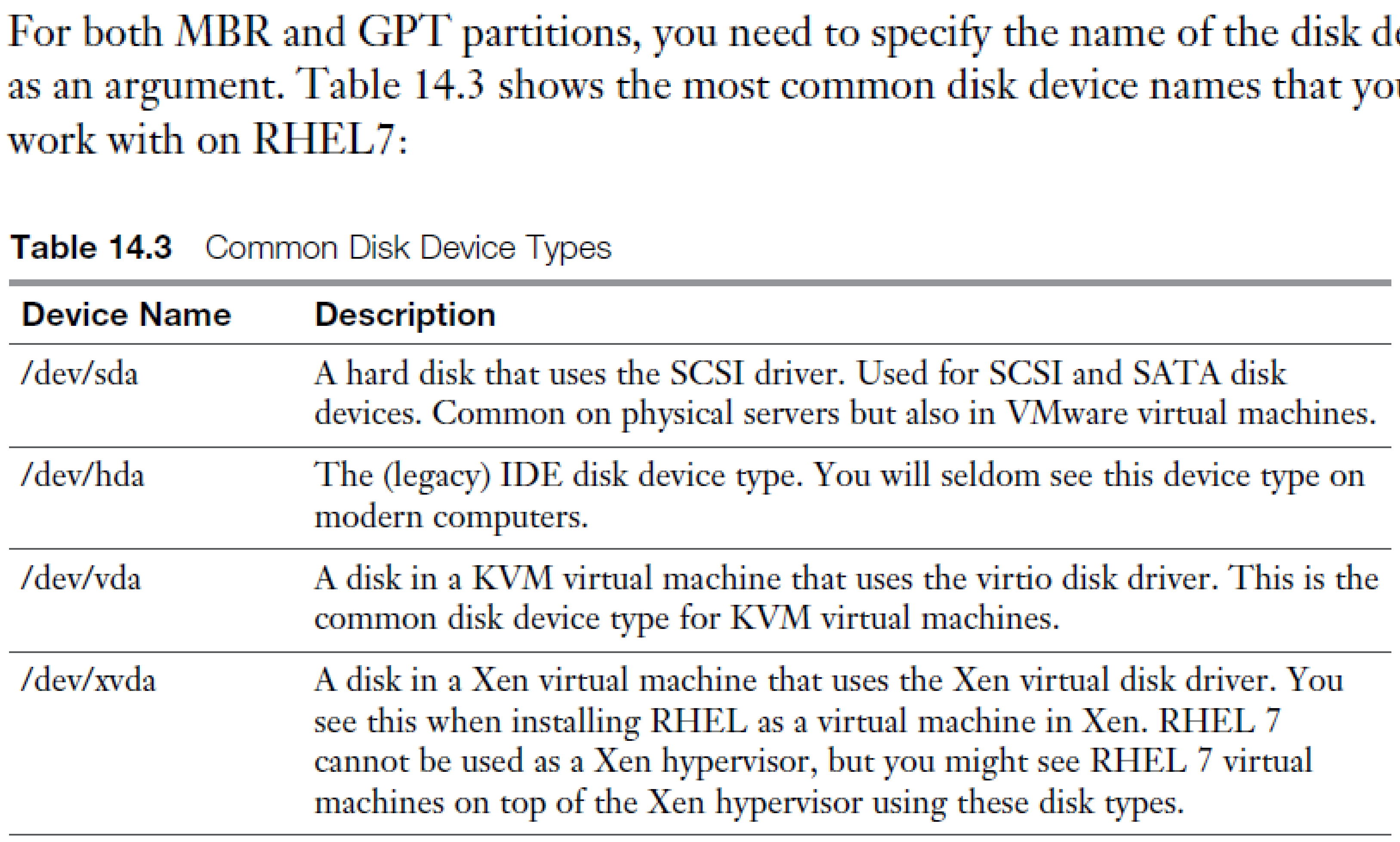
Understanding Storage Measurement Units

When talking about storage, different measurement units are used. In some cases, units like megabyte (MB) are used. In other cases, units like mebibyte (MiB) are used. The difference between these two is that a megabyte is a multiple of 1,000, and a Mebibyte is a multiple of 1,024. In computers, it makes sense to talk about multiples of 1,024 because that is how computers address items. The misunderstanding was created though when hardware vendors a long time ago started talking about megabytes instead.

Table 14.2 Disk Size Specifications

Symbol	Name	Value	Symbol	Name	Value
KB	Kilobyte	1000^1	KiB	Kibibyte	1024^1
MB	Megabyte	1000^2	MiB	Mebibyte	1024^2
GB	Gigabyte	1000^3	GiB	Gibibyte	1024^3
TB	Terabyte	1000^4	TiB	Tebibyte	1024^4
PB	Petabyte	1000^5	PiB	Pebibyte	1024^5
EB	Exabyte	1000^6	EiB	Exbibyte	1024^6
ZB	Zettabyte	1000^7	ZiB	Zebibyte	1024^7
YB	Yottabyte	1000^8	YiB	Yobibyte	1024^8

Managing Partitions and File Systems



For both MBR and GPT partitions, you need to specify the name of the disk device as an argument. Table 14.3 shows the most common disk device names that you work with on RHEL7:

Table 14.3 Common Disk Device Types

Device Name	Description
/dev/sda	A hard disk that uses the SCSI driver. Used for SCSI and SATA disk devices. Common on physical servers but also in VMware virtual machines.
/dev/hda	The (legacy) IDE disk device type. You will seldom see this device type on modern computers.
/dev/vda	A disk in a KVM virtual machine that uses the virtio disk driver. This is the common disk device type for KVM virtual machines.
/dev/xvda	A disk in a Xen virtual machine that uses the Xen virtual disk driver. You see this when installing RHEL as a virtual machine in Xen. RHEL 7 cannot be used as a Xen hypervisor, but you might see RHEL 7 virtual machines on top of the Xen hypervisor using these disk types.

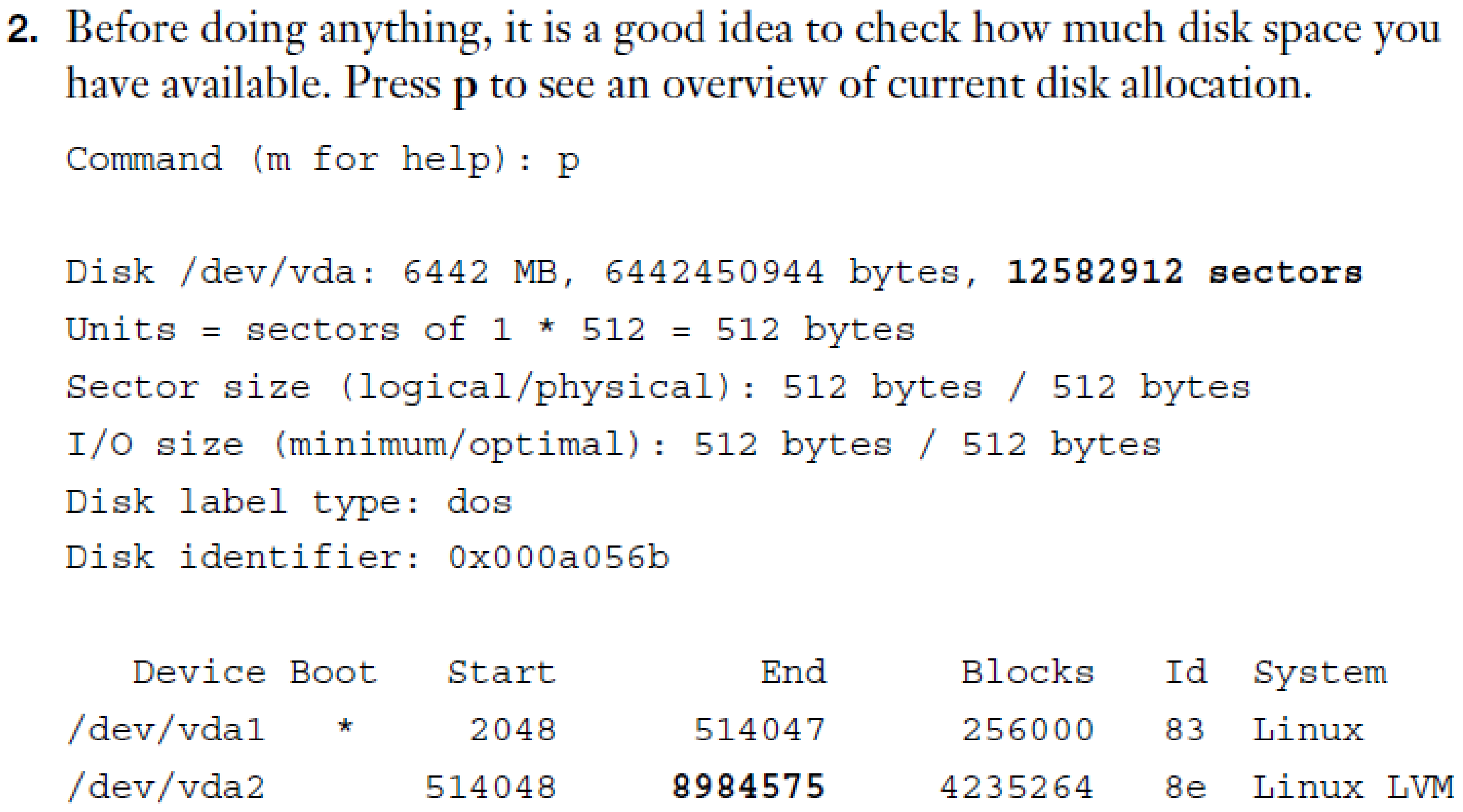
Exercise 14.1 Creating MBR Partitions with *fdisk*

1. Open a root shell and run the **fdisk** command. This command needs the name of the disk device where you want to create the partition as its argument. In this exercise I'll use /dev/vda. Change if needed according to your hardware.

   ```
   [root@localhost ~]# fdisk /dev/vda
   Welcome to fdisk (util-linux 2.23.2).
   ```

   Changes will remain in memory only until you decide to write them. Be careful before using the **write** command.

Exercise 14.1 Creating MBR Partitions with *fdisk*



2. Before doing anything, it is a good idea to check how much disk space you have available. Press **p** to see an overview of current disk allocation.

```
Command (m for help): p

Disk /dev/vda: 6442 MB, 6442450944 bytes, 12582912 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000a056b

   Device Boot    Start       End     Blocks   Id  System
/dev/vda1   *      2048     514047     256000   83  Linux
/dev/vda2        514048    8984575    4235264   8e  Linux LVM
```

Exercise 14.1 Creating MBR Partitions with *fdisk*

In the output of this command, in particular look for the total number of sectors and the last sector that is currently used (marked in bold in the above command output). If the last partition does not end on the last sector, you have available space to create a new partition.

3. Type **n** to add a new partition.

```
Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p):
```

Exercise 14.1 Creating MBR Partitions with *fdisk*

4. Assuming you have a /dev/vda1 and a /dev/vda2 partition and nothing else, select **p** to create a primary partition. Accept the partition number that is now suggested.
5. Specify the first sector on disk that the new partition will start on. The first available sector is suggested by default, press Enter to accept.
6. Specify the last sector that the partition will end on. By default, the last sector available on disk is suggested. If you use that, after this exercise you will not have any disk space left to create additional partitions or logical volumes, so you should use another last sector. To use another last sector, you can do the following:

Exercise 14.1 Creating MBR Partitions with *fdisk*

- Enter the number of the last sector you want to use.
- Enter **+number** to create a partition that sizes a specific number of sectors.
- Enter **+number(K,M,G)** to specify the size you want to assign to the partition in KiB, MiB, or GiB.
- Type **+100M** to make this a 100 MiB partition.

```
Partition number (3,4, default 3):
First sector (8984576-12582911, default 8984576):
Using default value 8984576
Last sector, +sectors or +size{K,M,G} (8984576-12582911,
default 12582911): +100M
Partition 3 of type Linux and of size 100 MiB is set
```

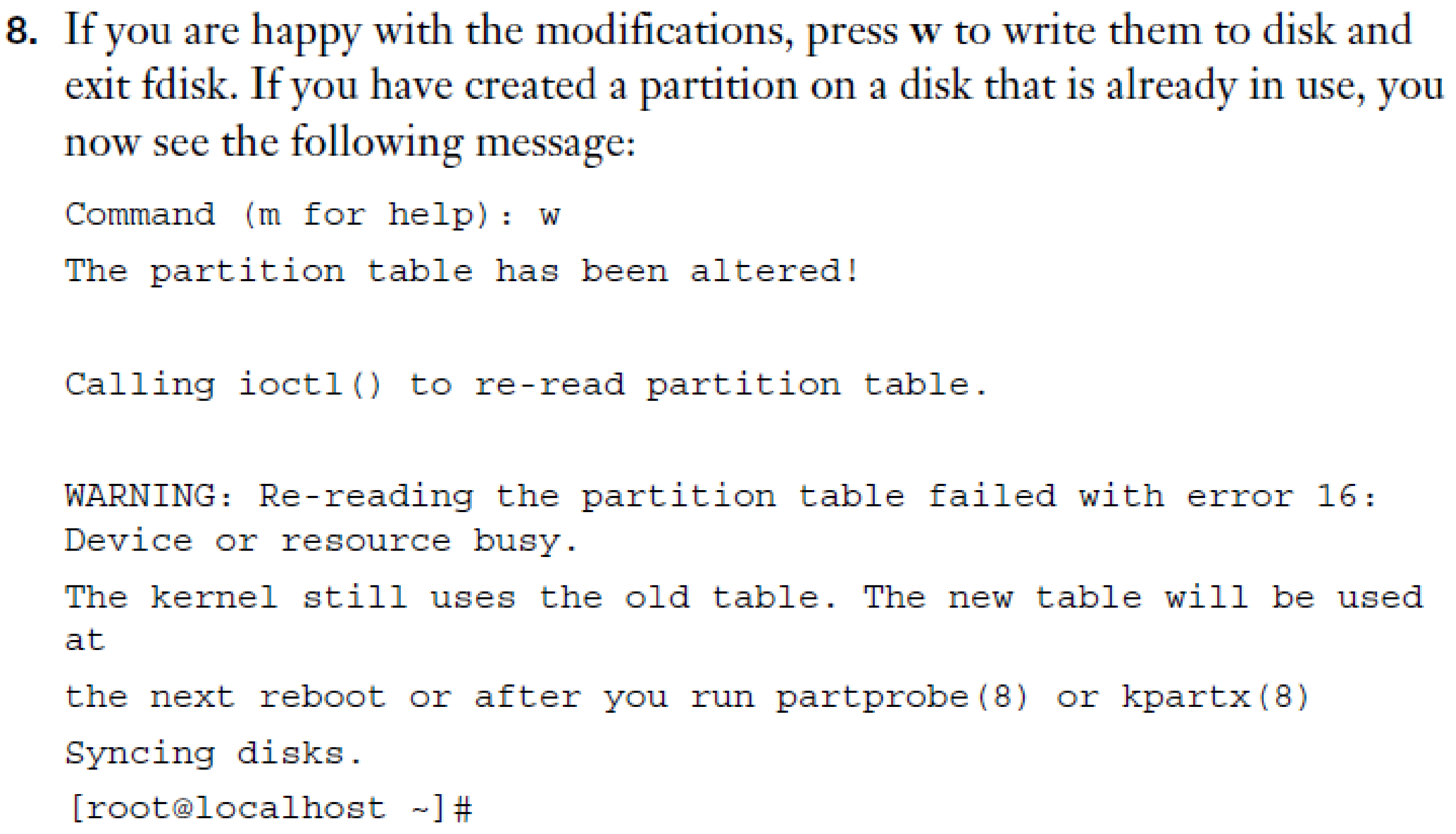
Exercise 14.1 Creating MBR Partitions with *fdisk*

After you enter the partition's ending boundary, fdisk will show a confirmation.

7. At this point, you can define the partition type. By default, a Linux partition type is used. If you want the partition to be of any other partition type, use **t** to change it. Common partition types include the following:
   - 82: Linux swap
   - 83: Linux
   - 8e: Linux LVM

Press **Enter** to accept the default Linux partition type **83**.

Exercise 14.1 Creating MBR Partitions with *fdisk*



8. If you are happy with the modifications, press **w** to write them to disk and exit fdisk. If you have created a partition on a disk that is already in use, you now see the following message:

```
Command (m for help): w
The partition table has been altered!


Calling ioctl() to re-read partition table.


WARNING: Re-reading the partition table failed with error 16:
Device or resource busy.
The kernel still uses the old table. The new table will be used
at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@localhost ~]#
```

Exercise 14.1 Creating MBR Partitions with *fdisk*

9. This message indicates that the partition has successfully been added to the partition table, but the in-memory kernel partition table could not be updated. You can see that by comparing the output of **fdisk -l /dev/vda** with the output of the command **cat /proc/partitions**, which shows the kernel partition table.

10. Type **partprobe /dev/vda** to write the changes to the kernel partition table. The partition has now been added and you can create a file system on it as described in the section "Creating File Systems."

NOTE You see the "re-reading the partition table failed with error 16" message only if you are adding partitions to a disk that already has some mounted partitions. If you are working on a new disk that does not have any mounted partitions, you will not see this error and you will not have to use the **partprobe** command.

TIP The **fdisk** utility writes changes to disk only when you enter **w**, which is the **fdisk** write command. If you have made a mistake and want to get out, press **q** to quit.

Creating GPT Partitions with *gdisk*

If a disk is configured with a GUID partition table (GPT), or if it is a new disk that does not contain anything yet and has a size that goes beyond 2 TiB, you need to use the gdisk utility to create partitions. This utility has a lot of similarities with fdisk but some differences as well. The following procedure shows how to create partitions in gdisk.

NOTE fdisk has some support for managing GPT partitions also. At the time of this writing, the GPT support in fdisk is not stable. For that reason, it is recommended to use gdisk on GPT partitions and fdisk on MBR partitions.

Exercise 14.2 Creating GPT Partitions with *gdisk*

1. To create a partition with gdisk, type **gdisk /dev/vdb**. (Replace /dev/vdb with the exact device name used on your computer.) Gdisk will try to detect the current layout of the disk, and if nothing has been detected, it will create the GPT partition table and associated disk layout.

```
[root@localhost ~]# gdisk /dev/vdb
GPT fdisk (gdisk) version 0.8.6

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries.

Command (? for help):
```

Exercise 14.2 Creating GPT Partitions with *gdisk*

2. Type **n** to enter a new partition. You can choose any partition number between 1 and 128, but it is wise to accept the default partition number that is suggested.

```
Command (? for help): n
Partition number (1-128, default 1):
```

3. You now are asked to enter the first sector. By default, the first sector that is available on disk will be used, but you can specify an offset as well. This does not make sense at all, so just press **Enter** to accept the default first sector that is proposed.

```
First sector (34-2097118, default = 2048) or {+-}size{KMGTP}:
```

Exercise 14.2 Creating GPT Partitions with *gdisk*

4. When asked for the last sector, by default the last sector that is available on disk is proposed (which would create a partition that fills the entire hard disk). You can specify a different last sector, or specify the disk size using +, the size, and KMGTP. So to create a 2 TiB disk partition, use **+2TiB**.

   ```
   Last sector (2048-2097118, default = 2097118) or {+-}size{KMGTP}:
   +100M
   ```

5. You now are asked to set the partition type. If you do not do anything, the partition type is set to 8300, which is the Linux file system partition type. Other options are available as well. You can press **l** to show a list of available partition types.

   The relevant partitions types are as follows:

   - **8200:** Linux swap
   - **8300:** Linux file system
   - **8e00:** Linux LVM

   Notice that these are the same partition types as the ones that are used in MBR, with two 0s added to their names. You can also just press **Enter** to accept the default partition type 8300.

Exercise 14.2 Creating GPT Partitions with *gdisk*

6. The partition is now created (but not yet written to disk). Press **p** to show an overview, which allows you to verify that this is really what you want to use.

```
Command (? for help): p
Disk /dev/vdb: 2097152 sectors, 1024.0 MiB
Logical sector size: 512 bytes
Disk identifier (GUID): 870DF067-6735-482E-83CE-5123E20509E0
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 2097118
Partitions will be aligned on 2048-sector boundaries
Total free space is 1892285 sectors (924.0 MiB)

Number  Start (sector)  End (sector)  Size       Code  Name
   1    2048            206847  100.0 MiB  8300  Linux filesystem
```

Exercise 14.2 Creating GPT Partitions with *gdisk*

7. If you are satisfied with the current partitioning, press **w** to write changes to disk and commit. This gives a warning, after which the new partition table is written to the GUID partition table.

```
Command (? for help): w


Final checks complete. About to write GPT data. THIS WILL
OVERWRITE EXISTING
PARTITIONS!!


Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/vdb.
The operation has completed successfully.
```

8. If at this point you get an error message indicating that the partition table is in use, type **partprobe** to update the kernel partition table.

Creating File Systems

At this point, you know how to create partitions. A partition all by itself is not very useful. It only becomes useful if you decide to do something with it. That often means that you have to put a file system on top of it. In this section, you learn how to do that.

Different file systems can be used on RHEL 7. Table 14.4 provides an overview of the most common file systems.

Creating File Systems

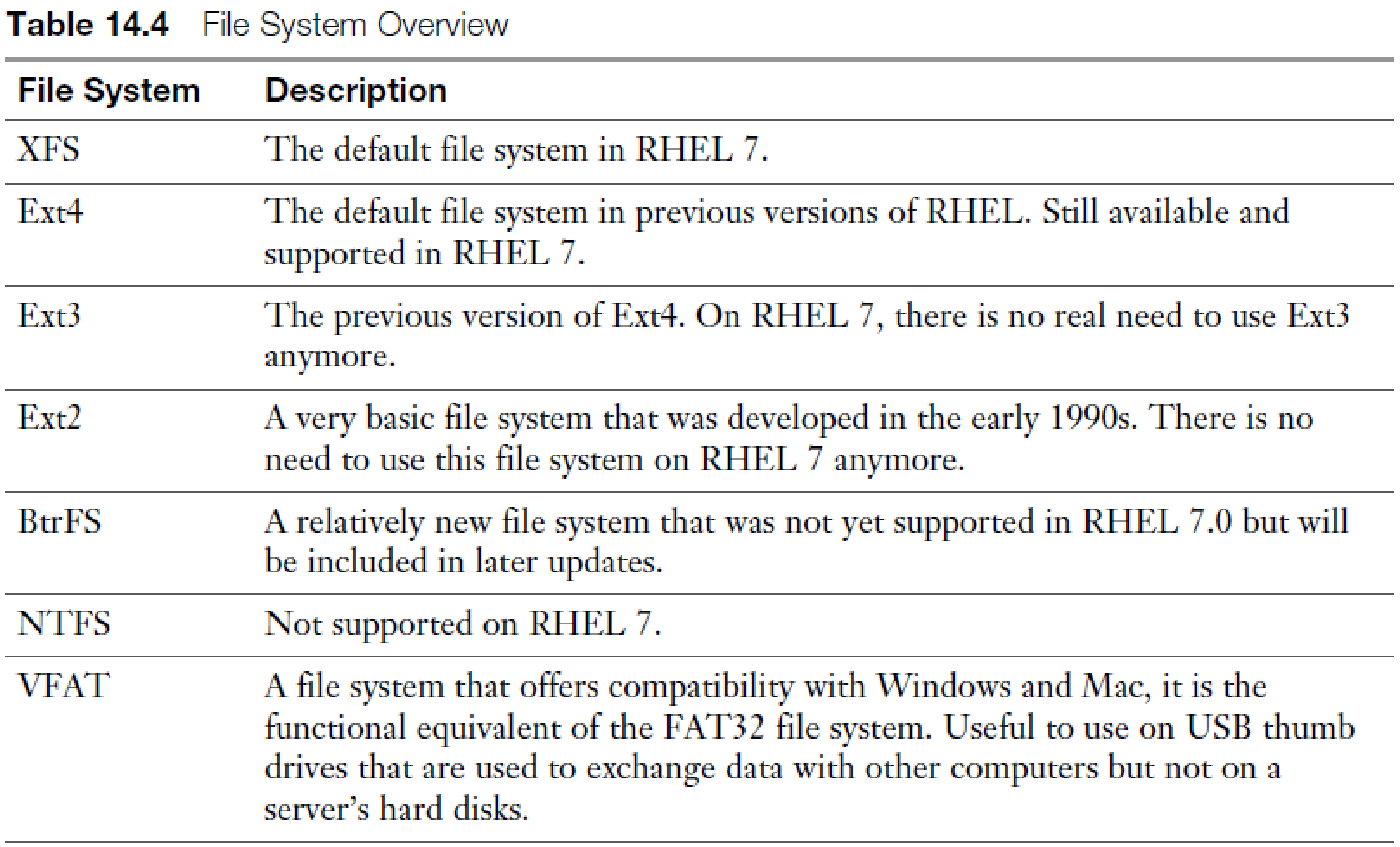


Table 14.4 File System Overview

File System	Description
XFS	The default file system in RHEL 7.
Ext4	The default file system in previous versions of RHEL. Still available and supported in RHEL 7.
Ext3	The previous version of Ext4. On RHEL 7, there is no real need to use Ext3 anymore.
Ext2	A very basic file system that was developed in the early 1990s. There is no need to use this file system on RHEL 7 anymore.
BtrFS	A relatively new file system that was not yet supported in RHEL 7.0 but will be included in later updates.
NTFS	Not supported on RHEL 7.
VFAT	A file system that offers compatibility with Windows and Mac, it is the functional equivalent of the FAT32 file system. Useful to use on USB thumb drives that are used to exchange data with other computers but not on a server's hard disks.

Creating File Systems

To format a partition with one of the supported file systems, you can use the **mkfs** command, using the option **-t** to specify which specific file system needs to be used. Alternatively, one of the file system specific tools can be used, such as mkfs.ext4 to format an Ext4 file system.

NOTE If you are using mkfs without any further specification of which file system you want to format, an Ext2 file system will be formatted. This is probably not what you want to use, so do not forget to specify which file system you want to use.

Listing 14.1 Formatting a File System with XFS

```
 [root@server3 ~]# mkfs -t xfs /dev/vda5
meta-data=/dev/vda5              isize=256    agcount=4, agsize=6400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0
data     =                       bsize=4096   blocks=25600, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

Changing File System Properties

When working with file systems, some properties can be managed as well. File system properties are specific for the file system you are using, so you work with different properties and different tools for the different file systems.

Managing Ext4 File System Properties

The generic tool for managing Ext4 file system properties is tune2fs. This tool was developed a long time ago for the Ext2 file system and is compatible with Ext3 and Ext4 also. When managing Ext4 file system properties, the **tune2fs -l** command is a nice command to start with. Listing 14.2 shows the output of this command where different file system properties are shown:

Managing Ext4 File System Properties

Listing 14.2 Showing File System Properties with **tune2fs -l**

```
[root@server1 ~]# tune2fs -l /dev/sdb1
tune2fs 1.42.9 (28-Dec-2013)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          2fbf3e73-166f-4a3d-8df6-ccfe727c1da5
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index
filetype extent 64bit flex_bg sparse_super huge_file uninit_bg dir_
nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
```

Adding Swap Partitions

You use most of the partitions on a Linux server for regular file systems. On Linux, swap space is normally also allocated on a disk device. That can be a partition, or an LVM logical volume (discussed in Chapter 15. In case of emergency, you can even use a file to extend the available swap space.

Using swap on Linux is a convenient way to improve Linux kernel memory usage. If a shortage of physical RAM occurs, non-recently-used memory pages can be moved to swap, which makes more RAM available for programs that need access to memory pages. Most Linux servers for that reason are configured with a certain amount of swap. If swap starts being used intensively, you could be in trouble though, and that is why swap usage should be closely monitored.

Exercise 14.5 Creating a Swap Partition

1. Use **fdisk /dev/vda** to open your disk in fdisk. (Use gdisk if you are using a disk with a GUID partition table.)
2. Press **n** to add a new partition. Specify start and stop cylinders and size.
3. Type **t** to change the partition type. If you are using fdisk, use partition type 82. If you are using gdisk, use partition type 8200.
4. Use mkswap to format the partition as swap space. Use, for instance, **mkswap /dev/vda6** if the partition you have just created is /dev/vda6.
5. Type **free -m**. You see the amount of swap space that is currently allocated.
6. Use swapon to switch on the newly allocated swap space. If, for instance, the swap device you have just created is /dev/vda6, use **swapon /dev/vda6** to activate the swap space.
7. Type **free -m** again. You see that the new swap space has been added to your server.

Adding Swap Files

If you do not have free disk space to create a swap partition and you do need to add swap space urgently, you can use a swap file as well. From a performance perspective, it does not even make that much difference if a swap file is used instead of a swap device such as a partition or a logical volume, and it may help you fixing an urgent need in a timely manner.

To add a swap file, you need to create the file first. The **dd if=/dev/zero of=/swapfile bs=1M count=100** command would add 100 blocks with a size of 1 Mebibyte from the /dev/zero device (which generates 0s) to the /swapfile file. The result is a 100 MiB file that can be configured as swap. To do so, you can follow the same procedure as for swap partitions. First use **mkswap /swapfile** to mark the file as a swap file, after which you can use **swapon /swapfile** to active it.

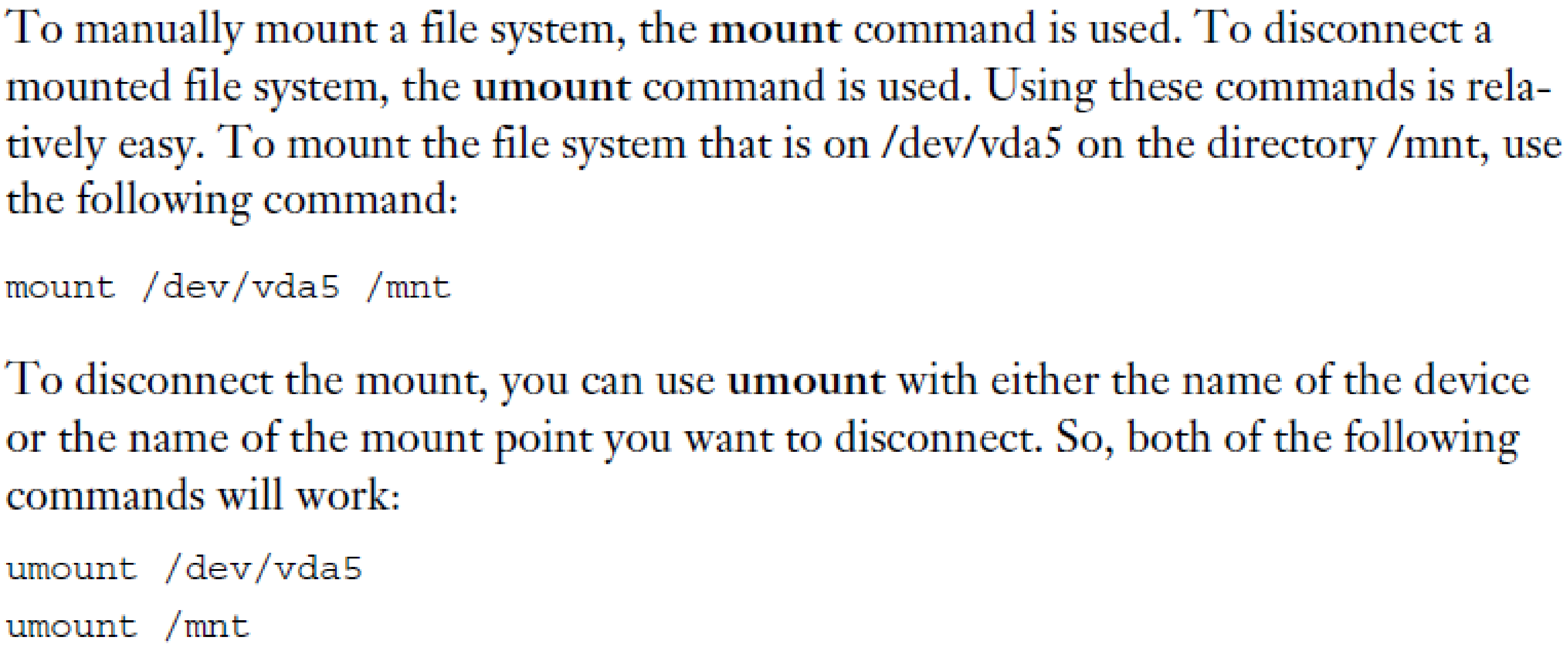
Mounting File Systems

Just creating a partition and putting a file system on it is not enough to start using it. To use a partition, you have to mount it as well. By mounting a partition (or better, the file system on it), you make its contents accessible through a specific directory.

To mount a file system, some information is needed:

- **What to mount:** This information is mandatory and specifies the name of the device that needs to be mounted.
- **Where to mount it:** This is also mandatory information which specifies the directory on which the device should be mounted.
- **What file system to mount:** Optionally, you can specify the file system type. In most cases, this is not necessary. The **mount** command will detect which file system is used on the device and make sure the correct driver is used.
- **Mount options:** Many mount options can be used when mounting a device. Using options is optional and depends on the needs you may have with the file system.

Manually Mounting File Systems



To manually mount a file system, the **mount** command is used. To disconnect a mounted file system, the **umount** command is used. Using these commands is relatively easy. To mount the file system that is on /dev/vda5 on the directory /mnt, use the following command:

```
mount /dev/vda5 /mnt
```

To disconnect the mount, you can use **umount** with either the name of the device or the name of the mount point you want to disconnect. So, both of the following commands will work:

```
umount /dev/vda5
umount /mnt
```

Using Device Names, UUIDs, or Disk Labels

To mount a device, the name of the device can be used, as in the command /dev/vda5. If your server is used in an environment where a dynamic storage topology is used, this is not always the best approach. You may today have a storage device /dev/sda5, which after changes in the storage topology can be /dev/sdb5 after the next reboot of your server. This is why on a default RHEL 7 installation UUIDs are used instead of device names.

Every file system by default has a UUID associated to it, not just file systems that are used to store files but also special file systems such as the swap file system. You can use the **blkid** command to get an overview of the current file systems on your system and the UUID that is used by that file system.

Using Device Names, UUIDs, or Disk Labels

Listing 14.3 Using **blkid** to Find Current File System UUIDs

```
[root@server3 ~]# blkid
/dev/vda1: UUID="02305166-840d-4f74-a868-c549b3229e65" TYPE="xfs"
/dev/vda2: UUID="Hasz5q-nZF3-XN94-L2fm-xUwz-3VEq-qVCAYi" TYPE="LVM2_
member"
/dev/vda5: UUID="42f419c4-633f-4ed7-b161-519a4dadd3da" TYPE="xfs"
/dev/mapper/centos-swap: UUID="5867ba02-fd89-475c-be56-7922febde43b"
TYPE="swap"
/dev/mapper/centos-root: UUID="b2022ac4-73c6-4e6b-a52f-703e3e2476b7"
TYPE="xfs"
```

Using Device Names, UUIDs, or Disk Labels

Before the use of UUIDs was common, file systems were often configured to work with labels, which can be set using the **e2label** or the **xfs_admin -L** commands. This has become more uncommon in recent Linux versions. If a file system has a label, the **blkid** command will also show it.

To mount a file system based on a UUID, you use **UUID=nnnnn** instead of the device name. So if you want to mount /dev/vda5 from Listing 14.3 based on its UUID, the command becomes as follows:

```
mount UUID="42f419c4-633f-4ed7-b161-519a4dadd3da" /mnt
```

Manually mounting devices using the UUID is not exactly easier. If mounts are automated as discussed in the next section, however, it does make sense using UUIDs instead of device names.

To mount a file system using a label, you use the **mount LABEL=labelname** command. For example, use **mount LABEL=mylabel /mnt** to temporarily mount the file system with the name mylabel on the /mnt directory.

Automating File System Mounts Through */etc/fstab*

Normally, you do not want to be mounting file systems manually. Once you are happy with them, it is a good idea to have them mounted automatically. The classical way to do this is through the /etc/fstab file. Listing 14.4 shows sample contents of this file.

Automating File System Mounts Through *\/etc/fstab*

Listing 14.4 Sample /etc/fstab file Contents

```
[root@server3 ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Fri Jan 16 10:28:41 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root /           xfs     defaults        1 1
UUID=02305166-840d-4f74 /           xfs     defaults        1 2
/dev/mapper/centos-swap swap        swap    defaults        0 0
```

Automating File System Mounts Through */etc/fstab*

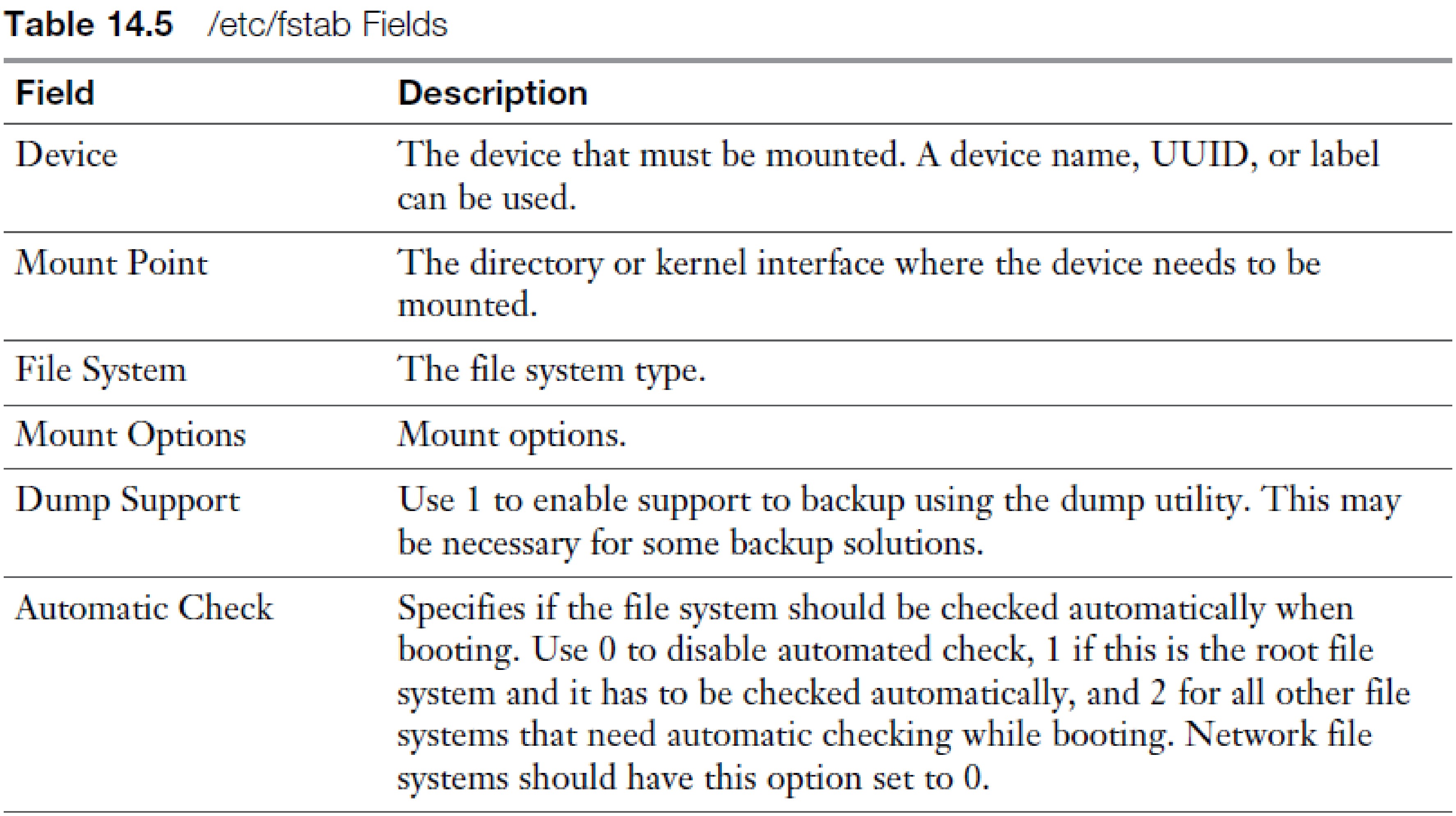
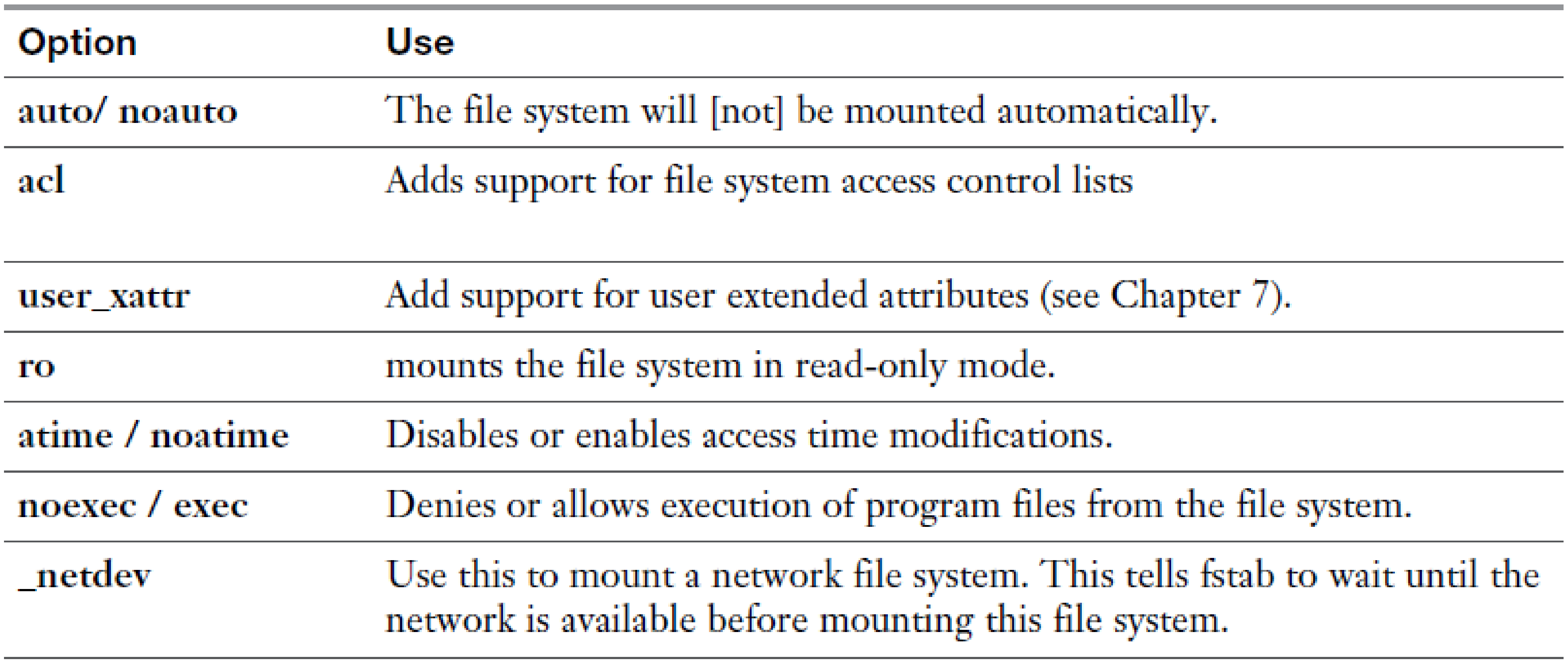


Table 14.5 /etc/fstab Fields

Field	Description
Device	The device that must be mounted. A device name, UUID, or label can be used.
Mount Point	The directory or kernel interface where the device needs to be mounted.
File System	The file system type.
Mount Options	Mount options.
Dump Support	Use 1 to enable support to backup using the dump utility. This may be necessary for some backup solutions.
Automatic Check	Specifies if the file system should be checked automatically when booting. Use 0 to disable automated check, 1 if this is the root file system and it has to be checked automatically, and 2 for all other file systems that need automatic checking while booting. Network file systems should have this option set to 0.

Automating File System Mounts Through */etc/fstab*

Table 14.6 Common Mount Options



Option	Use
auto/ noauto	The file system will [not] be mounted automatically.
acl	Adds support for file system access control lists
user_xattr	Add support for user extended attributes (see Chapter 7).
ro	mounts the file system in read-only mode.
atime / noatime	Disables or enables access time modifications.
noexec / exec	Denies or allows execution of program files from the file system.
_netdev	Use this to mount a network file system. This tells fstab to wait until the network is available before mounting this file system.

Automating File System Mounts Through */etc/fstab*

The fifth column of /etc/fstab specifies support for the dump utility. This is a utility that was developed to create file system backups. It is good practice to switch this feature on by specifying a 1 for all real file systems, and switch it off by specifying 0 for all system mounts.

The last column indicates if the file system integrity needs to be checked while booting. Put a 0 if you do not want to check the file system at all, a 1 if this is the root file system which needs to be checked before anything else, and a 2 if this is a nonroot file system that needs to be checked while booting.

Exercise 14.6 Mounting Partitions Through */etc/fstab*

1. From a root shell, type **blkid**. Use the mouse to copy the UUID="nnnn" part for /dev/vda5.
2. Type **mkdir -p /mounts/data** to create a mount point for this partition.
3. Open /etc/fstab in an editor and add the following line:

```
UUID="nnnn"         /mounts/data       xfs        defaults  1 2
```

4. Before attempting an automatic mount while rebooting, it is a good idea to test the configuration. Type **mount -a**. This mounts everything that is specified in /etc/fstab and that has not been mounted already.
5. Type **df -h** to verify that the partition has been mounted correctly.

Summary

In this important chapter, you learned how to work with partitions and file systems on RHEL 7. You learned how to create partitions for MBR and GPT disks, and how to put a file system on top of the partition. You also learned how to mount these partitions manually and automatically through /etc/fstab.

Define Key Terms

- Define the following key terms:

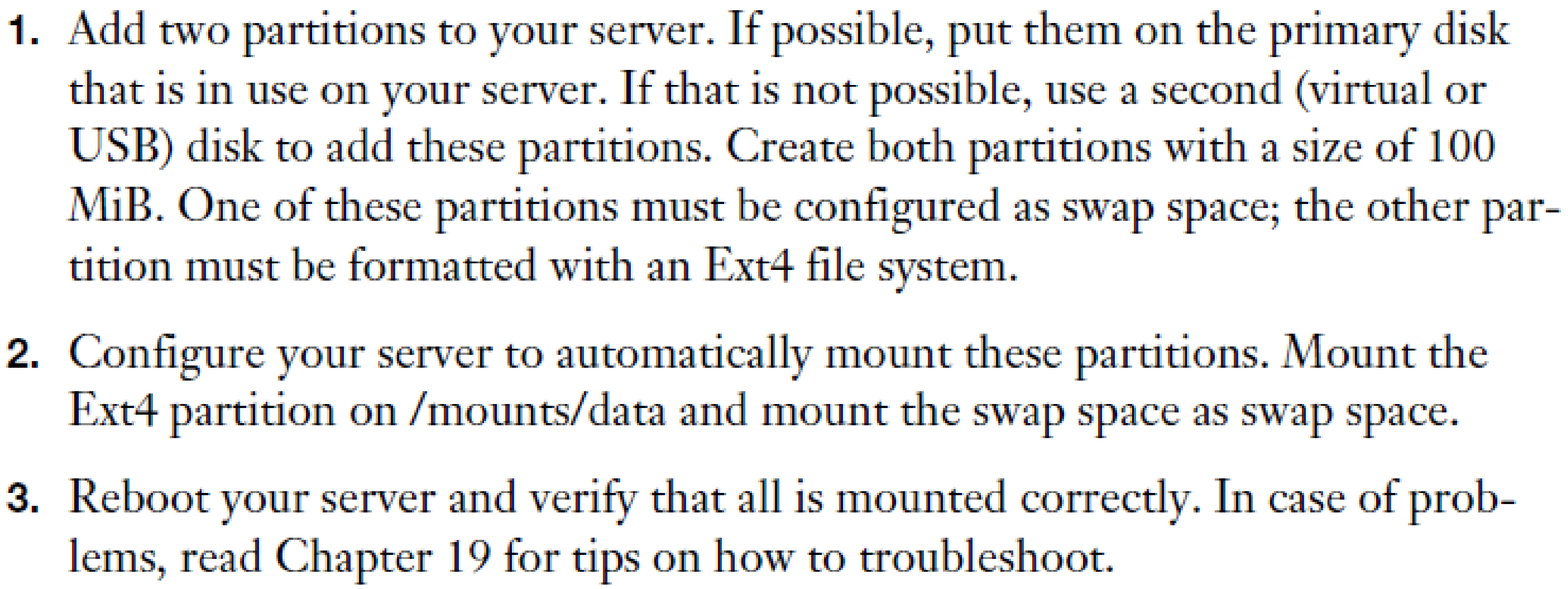
  - BIOS,
  - MBR,
  - partition,
  - primary partition,
  - extended partition,
  - logical partition,
  - GPT,
  - mount,
  - umount,
  - UUID,
  - label,
  - Ext2, Ext3, Ext4, XFS, BtrFS,VFAT,
  - fstab

Lab 14.1

1. Type **dd if=/dev/diskfile of=/dev/vda**. (Use **of=/dev/sda** if your disk device is /dev/sda instead of /dev/vda.)
2. Copy the backup of the /etc/fstab file, using **cp /root/fstab /etc**.

This restores the original configuration. You are now ready to start the end-of-chapter labs. After successfully completing the end-of-chapter labs, repeat this procedure. This allows you to work on clean disks when creating LVM logical volumes as described in the next chapter.

Lab 14.2



1. Add two partitions to your server. If possible, put them on the primary disk that is in use on your server. If that is not possible, use a second (virtual or USB) disk to add these partitions. Create both partitions with a size of 100 MiB. One of these partitions must be configured as swap space; the other partition must be formatted with an Ext4 file system.

2. Configure your server to automatically mount these partitions. Mount the Ext4 partition on /mounts/data and mount the swap space as swap space.

3. Reboot your server and verify that all is mounted correctly. In case of problems, read Chapter 19 for tips on how to troubleshoot.

Chapter 15:

Managing LVM Logical Volumes

Chapter 15 Objectives

- The following topics are covered in this chapter:
  - Understanding LVM
  - Creating LVM Logical Volumes
  - Resizing LVM Logical Volumes

- The following RHCSA exam objectives are covered in this chapter:
  - Create and remove physical volumes, assign physical volumes to volume groups, and create and delete logical volumes
  - Extend existing logical volumes

Managing LVM Logical Volumes

In the preceding chapter, you learned how to manage partitions on a hard disk. Creating multiple partitions on a disk is useful because it allows you to separate between different data types, but it does not offer the flexibility that the Logical Volume Manager (LVM) does. In this chapter, you learn how to configure and manage LVM logical volumes.

LVM Architecture

In the LVM architecture, several layers can be distinguished. On the lowest layer, the storage devices are used. These can be any storage devices, such as complete disks, partitions, logical units (LUNs) on a storage-area network (SAN) and whatever else is made possible in modern storage topologies.

The actual file systems are created on the logical volumes. As the logical volumes are flexible with regard to size, that makes the file systems flexible as well. If a file system is running out of disk space, it is relatively easy to extend the file system, or to reduce it if the file system allows that.

LVM Architecture

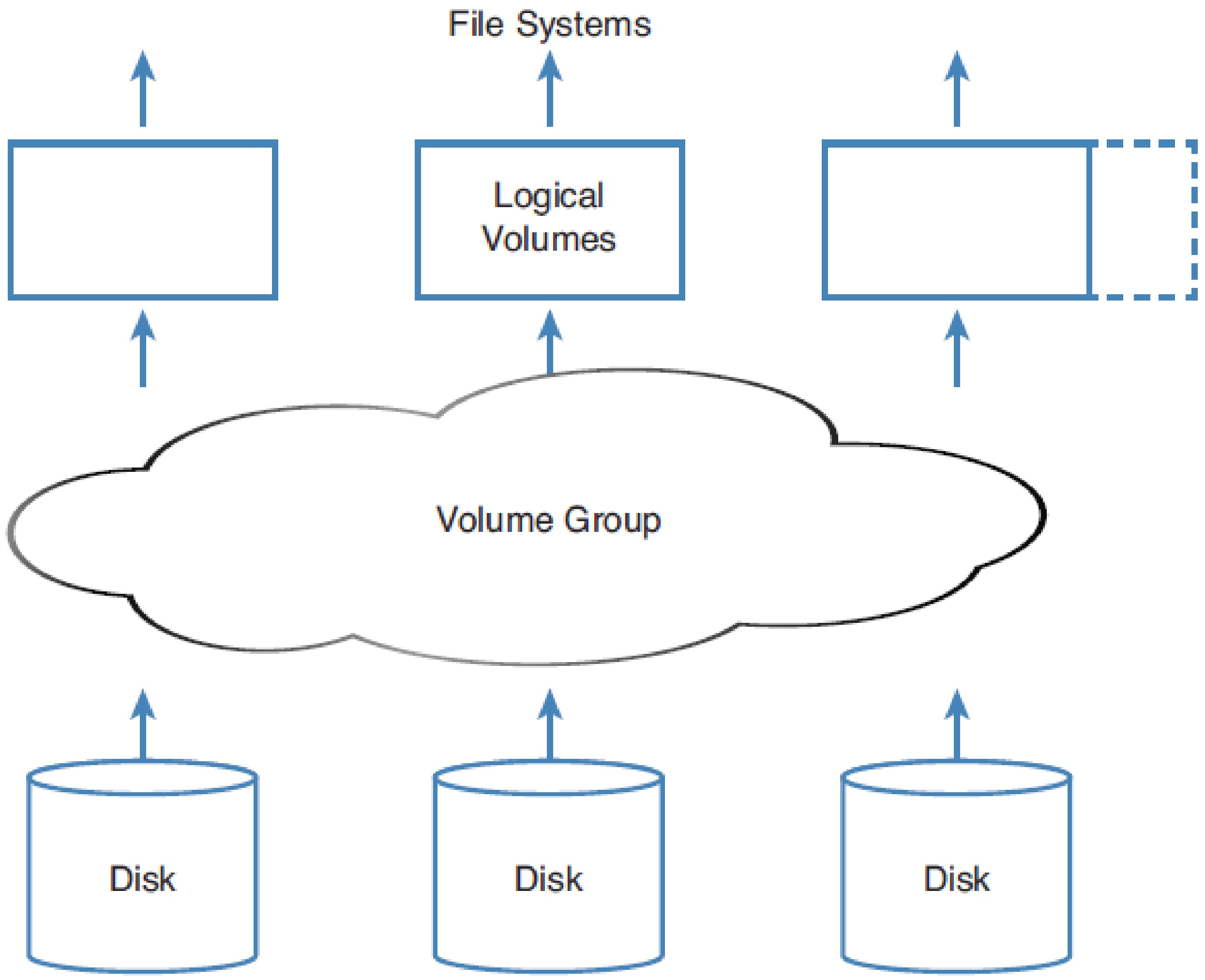
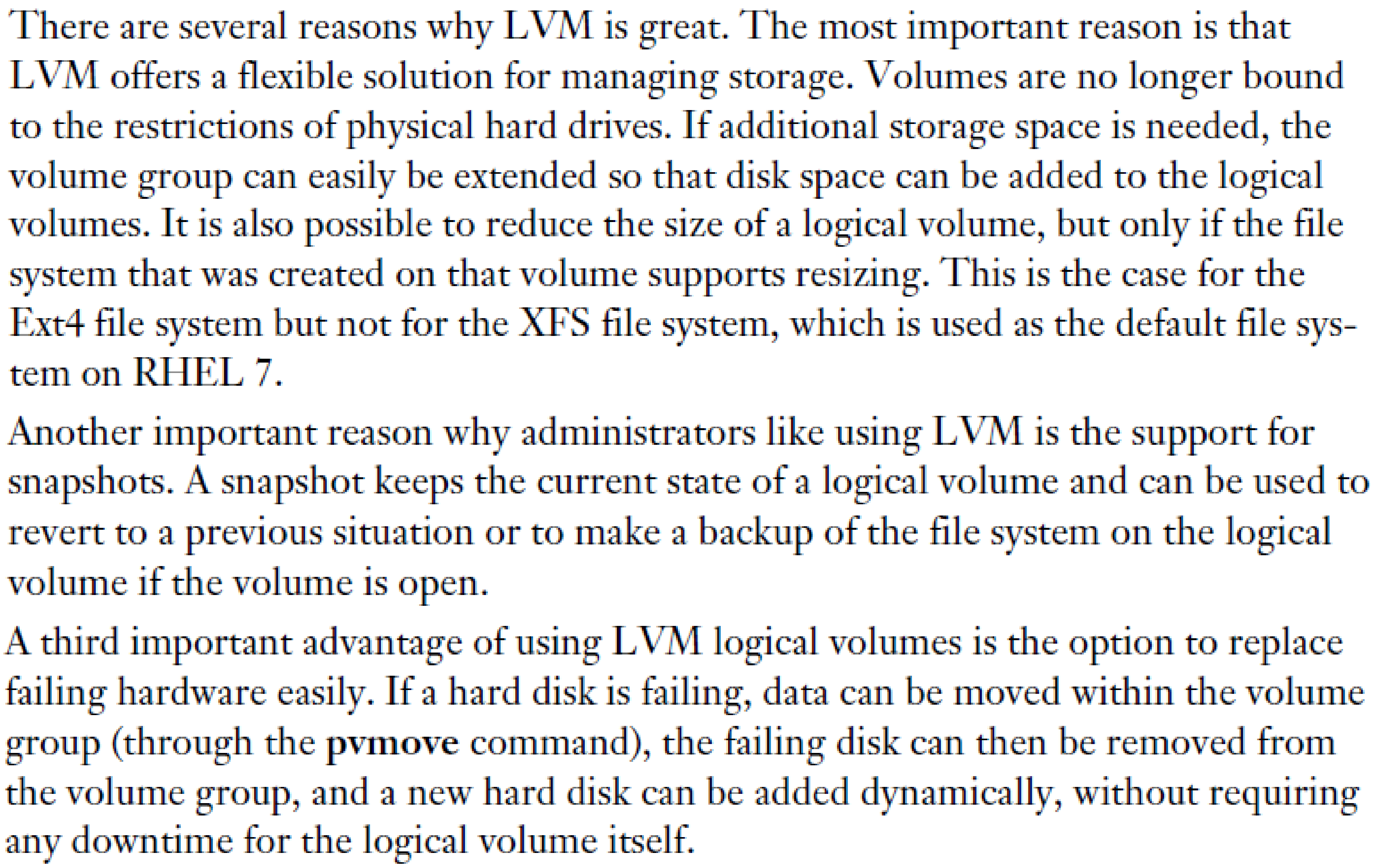


Figure 15.1 LVM architecture overview.

LVM Features



There are several reasons why LVM is great. The most important reason is that LVM offers a flexible solution for managing storage. Volumes are no longer bound to the restrictions of physical hard drives. If additional storage space is needed, the volume group can easily be extended so that disk space can be added to the logical volumes. It is also possible to reduce the size of a logical volume, but only if the file system that was created on that volume supports resizing. This is the case for the Ext4 file system but not for the XFS file system, which is used as the default file system on RHEL 7.

Another important reason why administrators like using LVM is the support for snapshots. A snapshot keeps the current state of a logical volume and can be used to revert to a previous situation or to make a backup of the file system on the logical volume if the volume is open.

A third important advantage of using LVM logical volumes is the option to replace failing hardware easily. If a hard disk is failing, data can be moved within the volume group (through the **pvmove** command), the failing disk can then be removed from the volume group, and a new hard disk can be added dynamically, without requiring any downtime for the logical volume itself.

Creating LVM Logical Volumes

Creating LVM logical volumes involves creating the three layers in the LVM architecture. You first have to take care of the physical volume (PV), then you need to create the volume group (VG) and assign physical volumes to it. As the last step, the logical volume (LV) itself has to be created. In this section, you learn what is involved in creating these three layers.

TIP You absolutely do not need to learn the commands discussed in this chapter by heart. All you really need to remember is **pv, vg, lv**. Open a command line, type **pv** and press the **Tab** key twice. This will show all commands that start with pv, which are all commands that are used for managing physical volumes. After you have found the command you need, run this command with the **--help** option. This shows a usage summary that lists everything that needs to be done to create the element you need. Listing 15.1 shows an example of the **pvcreate --help** command (which is explained in the next subsection).

Creating LVM Logical Volumes

Listing 15.1 Requesting help for the pvcreate command

```
[root@server1 ~]# pvcreate --help
  pvcreate: Initialize physical volume(s) for use by LVM

pvcreate
          [--norestorefile]
          [--restorefile file]
          [-d|--debug]
          [-f[f]|--force [--force]]
          [-h|-?|--help]
          [--labelsector sector]
          [-M|--metadatatype 1|2]
          [--pvmetadatacopies #copies]
          [--bootloaderareasize BootLoaderAreaSize[bBsSkKmMgGtTpPeE]]
          [--metadatasize MetadataSize[bBsSkKmMgGtTpPeE]]
          [--dataalignment Alignment[bBsSkKmMgGtTpPeE]]
          [--dataalignmentoffset AlignmentOffset[bBsSkKmMgGtTpPeE]]
          [--setphysicalvolumesize PhysicalVolumeSize[bBsSkKmMgGtTpPeE]
          [-t|--test]
          [-u|--uuid uuid]
          [-v|--verbose]
          [-y|--yes]
          [-Z|--zero {y|n}]
          [--version]
          PhysicalVolume [PhysicalVolume...]
```

Creating the Physical Volumes

Before the LVM tools can be used to create physical volumes, you need to create a partition marked as the LVM partition type. This is basically the same procedure as described in the preceding chapter, with the only difference that before writing changes to disk in fdisk or gdisk, you need to press **t** to change the partition type. (Exercise 15.1 shows exactly what you need to do.) If you are using an MBR disk, the partition type is 8e. If you are using a GUID disk, use the partition type 8300.

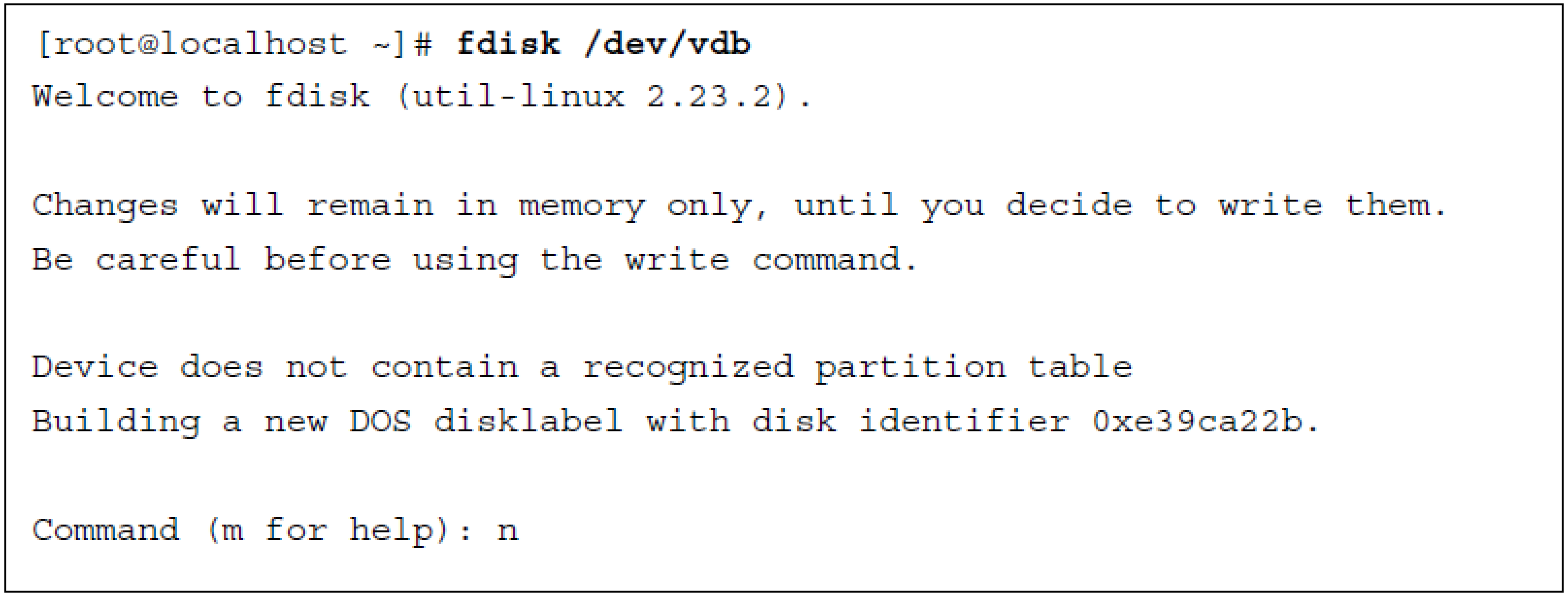
After creating the partition and flagging it as an LVM partition type, you need to use **pvcreate** to mark it as a physical volume. This writes some metadata to the partition, which allows it to be used in a volume group. The entire procedure is summarized in Exercise 15.1.

Exercise 15.1 Creating the Physical Volume

1. Open a root shell and type **fdisk /dev/vdb.**
2. Type **n** to create a new partition. Select **p** to make it a primary partition, and use the partition number that is suggested as a default. If you are using a clean device, this will be partition number 1.
3. Press **Enter** when asked for the first sector and type **+100M** to accept the last sector.
4. Once you are back on the fdisk prompt, type **t** to change the partition type. Because there is one partition only, fdisk does not ask which partition to use this partition type on. You may have to select a partition if you are using a different configuration.
5. The partitioner asks for the partition type you want to use. Type **8e**. Then, press **w** to write changes to disk and quit fdisk. Listing 15.2 shows an overview of all commands that have been used so far. If you are getting a message that the partition table could not be updated while writing the changes to disk, reboot your system.

Exercise 15.1 Creating the Physical Volume

Listing 15.2 Creating an LVM Partition in fdisk



```
[root@localhost ~]# fdisk /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0xe39ca22b.

Command (m for help): n
```

Exercise 15.1 Creating the Physical Volume

6. Now that the partition has been created, you need to flag it as an LVM physical volume. To do this, type **pvcreate /dev/vdb1**. You should now get this prompt: Physical volume "/dev/vbd1" successfully created.
7. Now type **pvs** to verify that the physical volume has been created successfully. The output may look like Listing 15.3. Notice that in this listing another physical volume already exists; that is because RHEL uses LVM by default to organize storage.

Listing 15.3 Verifying the Physical Volume

```
[root@localhost ~]# pvs
  PV         VG     Fmt  Attr PSize   PFree
  /dev/vda2  centos lvm2 a--    3.51g       0
  /dev/vdb1         lvm2 a--  100.00m 100.00m
```

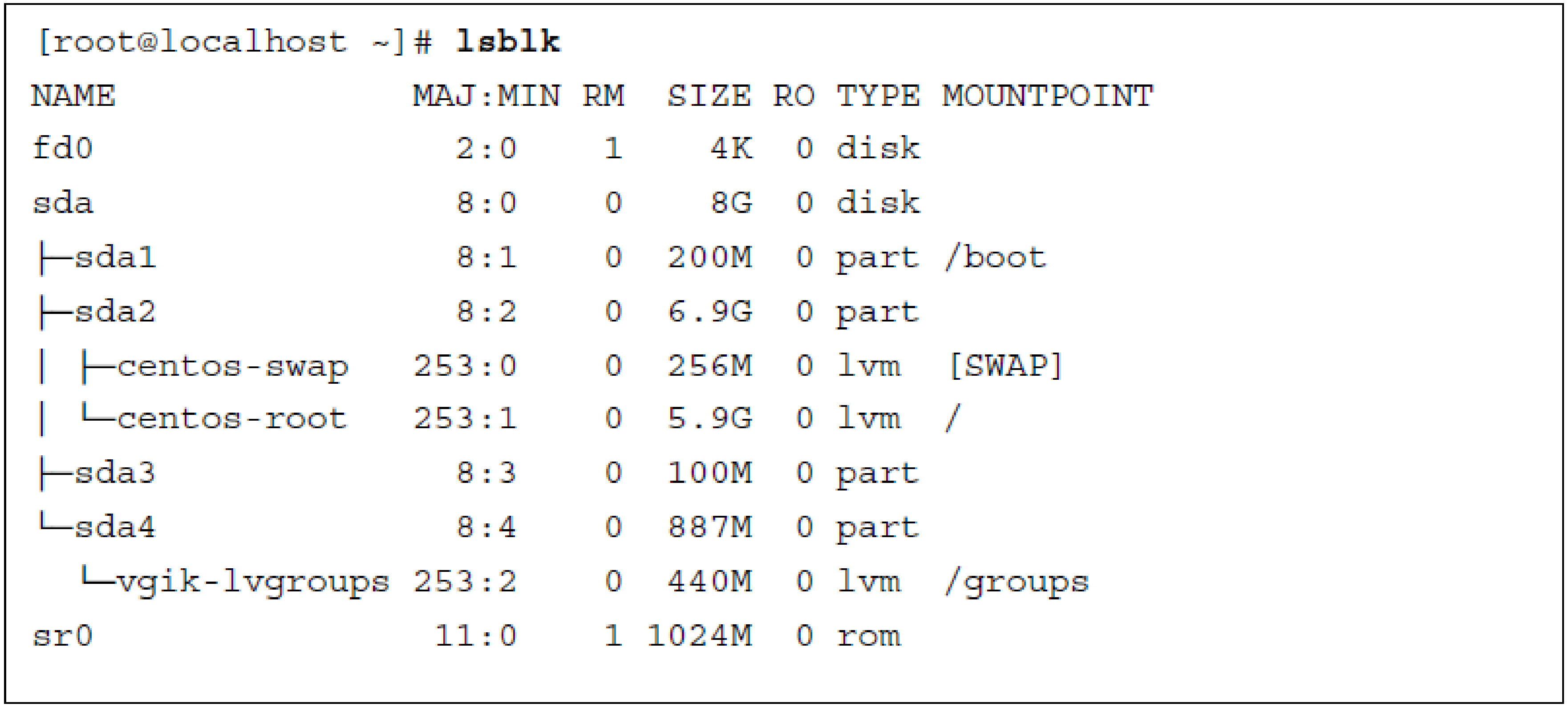
Creating the Physical Volume

Listing 15.4 Example **pvdisplay** Command Output

```
[root@server1 ~]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/vda2
  VG Name               centos
  PV Size               3.51 GiB / not usable 3.00 MiB
  Allocatable           yes (but full)
  PE Size               4.00 MiB
  Total PE              898
  Free PE               0
  Allocated PE          898
  PV UUID               CILii7-DzOd-w4L0-yOxi-9NXg-D3nP-ZugJIj
```

Creating the Physical Volume

Listing 15.5 Use **lsblk** for a Synthetic Overview of the Current Configuration of Storage on Your Server



```
[root@localhost ~]# lsblk
NAME                  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
fd0                     2:0    1     4K  0 disk
sda                     8:0    0     8G  0 disk
├─sda1                  8:1    0   200M  0 part /boot
├─sda2                  8:2    0   6.9G  0 part
│ ├─centos-swap       253:0    0   256M  0 lvm  [SWAP]
│ └─centos-root       253:1    0   5.9G  0 lvm  /
├─sda3                  8:3    0   100M  0 part
└─sda4                  8:4    0   887M  0 part
  └─vgik-lvgroups     253:2    0   440M  0 lvm  /groups
sr0                    11:0    1  1024M  0 rom
```

Creating the Volume Groups

Now that the physical volume has been created, you can assign it to a volume group. It is possible to add a physical volume to an existing volume group (which is discussed later in this chapter), but you will now learn how to create a new volume group and add the physical volume to it. This is a simple one-command procedure. Just type **vgcreate** followed by the name of the volume group you want to create and the name of the physical device you want to add to it. So, if the physical volume name is /dev/vdb1, the complete command is **vgcreate vgdata /dev/vdb1**.

In this procedure, you learned how to create a volume group in a two-step procedure where first the physical volume is created with the **pvcreate** command, after which the volume group is added using the **vgcreate** command. You can do this in a one-step procedure as well (where using a separate **pvcreate** command will not be necessary). If you are adding a partition to the volume group, however, it must be marked as partition type 8e already.

Creating the Volume Groups

When creating volume groups, a physical extent size is used. The physical extent size defines the size of the building blocks used to create logical volumes. A logical volume always has a size that is a multiple of the physical extent size. If you need to create huge logical volumes, it is more efficient to use a big physical extent size. If you do not specify anything, a default extent size of 4.00 MiB is used. The physical extent size is always specified as a multiple of 2 MiB, with a maximum size of 128 MiB. Use the **vgcreate -s** option to specify the physical extent size you want to use.

NOTE When working with LVM, there is the physical extent size to consider. This is the size of the basic building blocks used in the LVM configuration. When working with an ext4 file system, logical extents are used. The extent size on LVM are in no way related to the extent sizes that are used on the file systems.

After creating the volume group, you can request details about the volume group using the **vgs** command for a short summary, or the **vgdisplay** command to get more information. Listing 15.6 shows an example of the output of the **vgdisplay** command.

Creating the Volume Groups

Listing 15.6 Showing Current Volume Group Properties

```
[root@server1 ~]# vgdisplay
...

  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               248.00 MiB
  PE Size               4.00 MiB
  Total PE              62
  Alloc PE / Size       31 / 124.00 MiB
  Free  PE / Size       31 / 124.00 MiB
  VG UUID               TutZ0F-Fe0q-VGR4-y0dO-O0RM-Mv7e-09zyar
```

Creating the Logical Volumes and File Systems

Now that the volume group has been created, you can start creating logical volumes from it. This procedure is slightly more complicated than the creation of physical volumes or volume groups because there are more choices to be made. While creating the logical volume, you must specify a volume name and a size.

The volume size can be specified as an absolute value using the **-L** option. Use, for instance, **-L 5G** to create an LVM volume with a 5GB size. Alternatively, you can use relative sizes using the **-l** option. For instance, use **-l 50%FREE** to use half of all available disk space. You'll further need to specify the name of the volume group that the logical volume is assigned to, and optionally (but highly recommended), you can use **-n** to specify the name of the logical volume. For instance, use **lvcreate -n lvvol1 -L 100M vgdata** to create a logical volume with the name lvvol1 and add that to the vgdata volume group.

Understanding LVM Device Naming

Now that the logical volume has been created, you can start using it. To do this, you need to know the device name. LVM volume device names can be addressed in multiple ways. The simple method is to address the device as /dev/vgname/lvname. So if you have created a volume with the name lvdata, which gets its available disk space from the vgdata volume group, the device name would be /dev/vgdata/lvdata.

For naming LVM volumes, another system plays a role: device mapper. The device mapper (abbreviated as dm) is a generic interface that the Linux kernel uses to address storage devices. Device mapper is used by multiple device types, such as LVM volumes, but also by software RAID and advanced network devices such as multipath devices. These devices are created in two locations: as devices that are sequentially numbered in the /dev directory, such as /dev/dm-0, /dev/dm-1, and further. Because these device names do not provide any information about the device and therefore are confusing, symbolic links are created in the /dev/mapper directory. These symbolic links use a name that uses the vgname-lvname pattern. So, the device /dev/vgdata/lvdata would also be known as /dev/mapper/vgdata-lvdata. When working with LVM logical volumes, you can use any of these device names. Listing 15.7 shows an overview of the different LVM device names as provided by the device mapper.

Understanding LVM Device Naming

Listing 15.7 LVM Device Name Overview

```
[root@localhost ~]# \ls -l /dev/vgdata/lvdata
lrwxrwxrwx. 1 root root 7 Feb 11 05:00 /dev/vgdata/lvdata -> ../dm-2
[root@localhost ~]# \ls -l /dev/mapper/
total 0
lrwxrwxrwx. 1 root root       7 Feb 11 03:50 centos-root -> ../dm-1
lrwxrwxrwx. 1 root root       7 Feb 11 03:50 centos-swap -> ../dm-0
crw-------. 1 root root 10, 236 Feb 11 03:50 control
lrwxrwxrwx. 1 root root       7 Feb 11 05:00 vgdata-lvdata -> ../dm-2
[root@localhost ~]# \ls -l /dev/dm-2
brw-rw----. 1 root disk 253, 2 Feb 11 05:00 /dev/dm-2
```

Exercise 15.2 Creating the Volume Group and Logical Volumes

1. Open a root shell. Type **pvs** to verify the availability of physical volumes on your machine. You should see the /dev/vdb1 physical volume that was created previously.
2. Type **vgcreate vgdata /dev/vdb1**. This will create the volume group with the physical volume assigned to it.
3. Type **vgs** to verify that the volume group was created successfully. Also type **pvs**. Notice that this command now shows the name of the physical volumes, with the names of the volume groups they are assigned to.
4. Type **lvcreate -n lvdata -l 50%FREE vgdata**. This creates an LVM logical volume with the name lvdata, which will use 50% of available disk space in the vgdata volume group.

Exercise 15.2 Creating the Volume Group and Logical Volumes

5. Type **lvs** to verify that the volume was added successfully.
6. At this point, you are ready to create a file system on top of the logical volume. Type **mkfs.xfs /dev/vgdata/lvdata** to create the file system.
7. Type **mkdir /files** to create a folder on which the volume can be mounted.
8. Add the following line to /etc/fstab:

```
/dev/vgdata/lvdata /files    xfs       defaults  1 2
```

9. Type **mount -a** to verify that the mount works and mount the file system.

LVM Management Essential

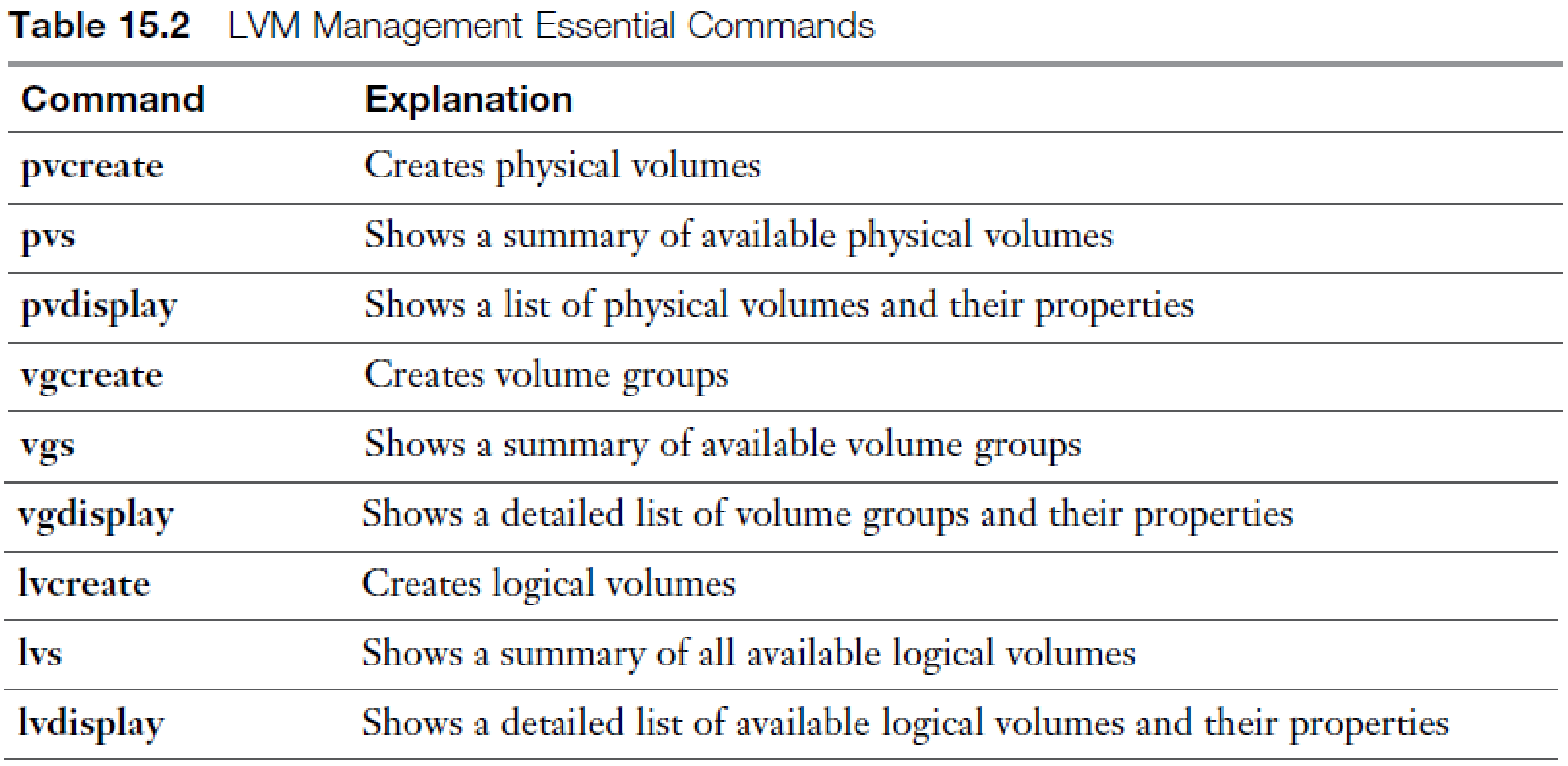


Table 15.2 LVM Management Essential Commands

Command	Explanation
pvcreate	Creates physical volumes
pvs	Shows a summary of available physical volumes
pvdisplay	Shows a list of physical volumes and their properties
vgcreate	Creates volume groups
vgs	Shows a summary of available volume groups
vgdisplay	Shows a detailed list of volume groups and their properties
lvcreate	Creates logical volumes
lvs	Shows a summary of all available logical volumes
lvdisplay	Shows a detailed list of available logical volumes and their properties

Resizing LVM Logical Volumes

One of the major benefits of using LVM is that LVM volumes are easy to resize, which is very useful if your file system is running out of available disk space. If the XFS file system is used, a volume can be increased, but not decreased, in size. Other file systems such as Ext4 and Btrfs support decreasing of the file system size also. Decreasing an Ext4 file system can be done offline only, which means that you need to unmount it before you can resize it. In this section, you learn how to increase the size of an LVM logical volume.

Resizing Volume Groups

The main part of LVM flexibility sits in the fact that it is so easy to resize the volume groups and the logical volumes that are using disk space from the volume group. The **vgextend** command is used to add storage to a volume group, and the **vgreduce** command is used to take physical volumes out of a volume group (which can lead to some additional complications). For the RHCSA test, you need to know how to extend the available storage in volume groups. This procedure is relatively easy:

1. Make sure that a physical volume or device is available to be added to the volume group.
2. Use **vgextend** to extend the volume group. The new disk space will show immediately in the volume group.

After extending a volume group, you can use the **vgs** command to verify that a physical volume has been added to the volume group. In Listing 15.8, you can see that the centos VG contains two physical volumes, as indicated in the #PV column.

Resizing Volume Groups

Listing 15.8 Verifying VG Resize Operations with **vgs**

```
[root@server2 ~]# vgs
  VG      #PV #LV #SN Attr   VSize   VFree
  centos    2   3   0 wz--n-   7.00g 560.00m
  vgsan     1   2   0 wz--n- 496.00m  96.00m
```

Resizing Logical Volumes and File Systems

Like volume groups can be extended with the **vgextend** command, logical volumes can be extended with the **lvextend** command. This command has a very useful option to take care of extending the file systems on the logical volume at the same time; it is recommended to use this option and not the alternative approach where logical volumes and the file systems on top of the logical volumes are extended separately. When resizing a logical volume with the file system it contains, nothing will happen to the file system, and its data will remain intact. Most file system resizing operations can even be done online, without any need to unmount the file system.

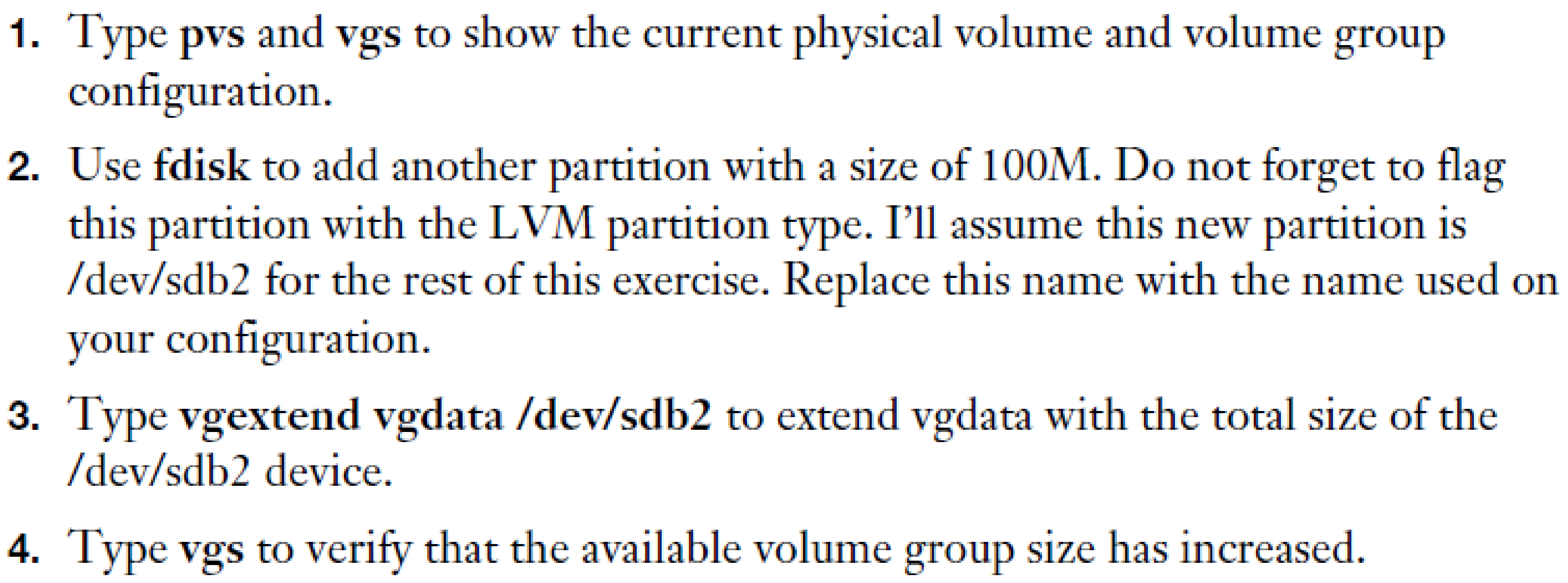
To grow the logical volume size, use **lvresize**, followed by the **-r** option to resize the file system used on it. Then, specify the size you want the resized volume to be. The easiest and most intuitive way to do that is by using **-L** followed by a **+** sign and the amount of disk space you want to add, as in **lvresize -L +1G -r /dev/vgdata/lvdata**. An alternative way to resize the logical volume is by using the **-l** option. This option is followed by the number of extents that are added to the logical volume or by the absolute or relative percentage of extents in the volume group that will be used. You can, for example, use the following commands to resize the logical volume:

Resizing Logical Volumes and File Systems

- **lvresize -r -l 75%VG /dev/vgdata/lvdata** This resizes the logical volume so that it will take 75% of the total disk space in the volume group.
- **lvresize -r -l +75%VG /dev/vgdata/lvdata** This tries to add 75% of the total size of the volume group to the logical volume. (Notice the difference with the previous command.)
- **lvresize -r -l +75%FREE /dev/vgdata/lvdata** This adds 75% of all free disk space to the logical volume.
- **lvresize -r -l 75%FREE /dev/vgdata/lvdata** This resizes the logical volume to a total size that equals 75% of the amount of free disk space. (Notice the difference with the previous command.)

NOTE The size of an XFS file system cannot be decreased; it can only be increased. If you need a file system that can be shrunk in size, use Ext4, not XFS.

Exercise 15.3 Resizing Logical Volumes



1. Type **pvs** and **vgs** to show the current physical volume and volume group configuration.
2. Use **fdisk** to add another partition with a size of 100M. Do not forget to flag this partition with the LVM partition type. I'll assume this new partition is /dev/sdb2 for the rest of this exercise. Replace this name with the name used on your configuration.
3. Type **vgextend vgdata /dev/sdb2** to extend vgdata with the total size of the /dev/sdb2 device.
4. Type **vgs** to verify that the available volume group size has increased.

Exercise 15.3 Resizing Logical Volumes

5. Type **lvs** to verify the current size of the logical volume lvdata.
6. Type **df -h** to verify the current size of the file system on lvdata.
7. Type **lvextend -r -l +50%FREE /dev/vgdata/lvdata** to extend lvdata with 50% of all available disk space in the volume group.
8. Type **lvs** and **df -h** again to verify that the added disk space has become available.
9. Type **lvreduce -r -L -150M /dev/vgdata/lvdata**. This shrinks the lvdata volume with 50MB. Notice that while doing this the volume is temporarily unmounted, which happens automatically.

Summary

In this chapter, you learned how to work with LVM logical volumes. First you learned how LVM provides features that are not available in normal partitioning, such as easy resizing or working with snapshots. You also learned how to create partitions that can be used in an LVM configuration, and you learned how to create physical volumes, volume groups, and logical volumes to create an LVM infrastructure. You also learned how to resize volume groups and logical volumes.

Define Key Terms

- Define the following key terms:

  - PV,
  - physical volume,
  - VG,
  - volume group,
  - LV,
  - logical volume,
  - device mapper,
  - physical extent,
  - logical extent,
  - snapshot

Lab 15.1

1. Create a 500MB logical volume lvgroup. Format it with the XFS file system and mount it persistently on /groups. Reboot your server to verify that the mount works.
2. After rebooting, add another 250MB to the lvgroup volume that you just created. Verify that the file system resizes as well while resizing the volume.
3. Verify that the volume extension was successful.

Q&A