7 Package Management

CERTIFICATION OBJECTIVES

Copyright © 2011 by The McGraw-Hill Companies

After installation is complete, after systems are secured, filesystems are administered, and more, you still have work to do. To customize the system as needed, you may need to add or remove packages, among other tasks. To make sure the right updates are installed, you need to know how to get a system working with the Red Hat Network (RHN) or the repository associated with a rebuild distribution.

To accomplish these tasks, you need to understand how to use the **rpm** and **yum** commands in detail. While they're "just" two commands, they are rich in detail. Entire books have been dedicated to the **rpm** command, such as the *Red Hat RPM Guide*, by Eric Foster-Johnson. For many, that degree of in-depth knowledge of the **rpm** command is no longer necessary, given the capabilities of the **yum** command, and the additional package management tools provided on RHEL 6.

INSIDE THE EXAM

Administrative Skills

As the management of RPM packages is a fundamental skill for Red Hat administrators, it's reasonable to expect to use the **rpm**, **yum**, and related commands on the RHCSA exam. In fact, the RHCE exam effectively assumes knowledge of such commands and more as effective prerequisite skills. The RHCSA objectives include two requirements addressed in this chapter:

- Install and update software packages from the RHN, a remote repository, or the local filesystem.
- Update the kernel package appropriately to ensure a bootable system.

A couple of other, closely related objectives are covered in other chapters. A predecessor to the RPM package are the compressed archives described in Chapter 9. An RHCE skill covered in Chapter 12 supports the configuration of an RPM package for "a single file."

Now let's break down these skills a bit. If you don't have access to the RHN, don't be intimidated. When you use a **yum** command to install and update packages from the RHN, it's no different from the commands that you would use to install and update packages from a remote repository.

Copyright © 2011 by The McGraw-Hill Companies

CERTIFICATION OBJECTIVE 7.01

The Red Hat Package Manager

One of the major duties of a system administrator is software management. New applications are installed. Services are updated. Kernels are patched. Without the right tools, it can be difficult to figure out what software is on a system, what is the latest update, and what applications depend on other software. Worse, you may install a new software package only to find it has overwritten a crucial file from a currently installed package.

The Red Hat Package Manager (RPM) was designed to alleviate these problems. With RPM, software is managed in discrete *packages*. An RPM package includes the software with instructions for adding, removing, and upgrading those files. When properly used, the RPM system can back up key configuration files before proceeding with upgrades and removals. It can also help you identify the currently installed version of any RPM-based application.

RPMs and the **rpm** command are far from ideal, which is why it has been supplemented with the **yum** command. With a connection to a repository such as that available from the RHN or third-party "rebuilds" such as Scientific Linux, you'll be able to use **yum** to satisfy dependencies.

What Is a Package?

In the generic sense, an RPM package is a container of files. It includes the group of files associated with a specific program or application, which normally includes binary installation scripts as well as configuration and documentation files. It also includes instructions on how and where these files should be installed and uninstalled.

An RPM package name usually includes the version, the release, and the architecture for which it was built. For example, the fictional penguin-3.4.5-26.x86_64.rpm package is version 3.4.5, build 26, and the x86_64 indicates that it is suitable for computers built to the AMD/Intel 64-bit architecture.

Many RPM packages are CPU-specific. You can identify the CPU type for the system with the* uname -i *or* uname -p *commands. More information is available from the contents of the /proc/cpuinfo file.

Copyright © 2011 by The McGraw-Hill Companies

What Is a Red Hat Package?

At the heart of this system is the RPM database. Among other things, this database tracks the version and location of each file in each RPM. The RPM database also maintains an MD5 checksum of each file. With the checksum, you can use the **rpm -V *package*** command to determine whether any file from that RPM package has changed. The RPM database makes it easy to add, remove, and upgrade individual packages, as it's configured to know which files to handle and where to put them.

RPM also manages conflicts between packages. For example, assume you have two different packages that use configuration files with the same name. Call the original configuration file /etc/*someconfig.conf*. You've already installed package X. If you then try to install an update package Y, RPM packages are designed to back up the original /etc/*someconfig.conf* file (with a filename like /etc/*someconfig.conf.rpmsave*) before installing package Y.

While RPM upgrades are supposed to preserve or save existing configuration files, there are no guarantees, especially if the RPM is designed by someone other than Red Hat. It's best to back up all applicable configuration files before upgrading any associated RPM package.

What Is a Repository?

RPM packages are frequently organized into repositories. Generally, such repositories include groups of packages with different functions. When downloading a package, you may need to navigate to a different directory, depending on the function of the repository. For example, the RHN includes the following RHEL 6 Server repositories (additional repositories are available):

- **Red Hat Enterprise Linux Server** The main repository, which includes both the packages associated with the original installation of RHEL 6, along with updates.
- **RHN Tools** A repository associated with the management of RHN connections, along with virtualization tools and utilities for automating Kickstart installation on virtual machines.
- **RHN Server Supplementary** A collection of packages released under licenses other than open source, such as Java and Acrobat reader.
- **RHEL V2VWIN** A single package that includes support for reading Microsoft-formatted partitions on RHEL virtual machines.

Copyright © 2011 by The McGraw-Hill Companies

- **RHEL Server Optional** A large group of packages not normally associated with server systems. It includes desktop packages and more.

In contrast, the repository categories for third-party rebuild distributions vary. Generally, they include categories such as main and extras, along with supplementary packages and more. In most cases, while the main repository includes just the packages available from the released DVD, updated packages are configured in their own repositories.

Each repository includes a database of packages in a repodata/ subdirectory. That database includes information on each package, including dependencies. That database allows installation requests to that directory to include all dependencies. If you have a subscription to the RHN, access to these repositories is enabled in the rhnplugin.conf file, in the /etc/yum/pluginconf.d directory. That file is discussed later in this chapter.

Later in this chapter, you'll examine how to configure connections to repositories with the configuration files associated with the **yum** command.

on the job

A dependency is a package that needs to be installed to make sure all the features of a target package is available.

Install an RPM Package

There are three basic commands *that may* install an RPM. They won't work if there are dependencies (packages that need to be installed first). For example, if you haven't installed the SELinux policy analysis command line tools package (setools-console) and try to install the SELinux configuration tool package (policycoreutils-gui), you'll get the following message (version numbers may vary):

```
# rpm -i policycoreutils-gui-2.0.83-19.1.el6.x86_64.rpm
error: Failed dependencies:
        policycoreutils-python = 2.0.83-19.1.el6 is needed by policycoreutils-gui-
2.0.83-19.1.el6.x86_64
        setools-console is needed by policycoreutils-gui-2.0.83-19.1.el6.x86_64
```

One way to test this is to mount the RHEL 6 DVD with the **mount /dev/cdrom /media** command. Next, find the noted policycoreutils-gui package in the Packages/ subdirectory. Alternatively, you could download that package directly from the Red Hat Network or a configured repository with the **yumdownloader policycoreutils-gui** command. This and other **yum** commands are discussed later in this chapter. Just be aware, some Linux GUI desktop environments automatically mount CD/DVD

Copyright © 2011 by The McGraw-Hill Companies

media that is inserted into associated drives. If so, you'll see the mount directory in the output to the **mount** command.

When dependency messages are shown, **rpm** does not install the given package. Note the dependency messages. First, it requires a policycoreutils-python package of the same version number, and a less-defined setools-console package.

on the Job

Sure, you can use the --force option to make rpm ignore dependencies, but that can lead to other problems, unless you install those dependencies as soon as possible. The best option is to use an appropriate yum command, described later in this chapter. In this case, a yum install policycoreutils-gui command would automatically install the other dependent RPM as well.

If you're not stopped by dependencies, there are three basic commands that can install RPM packages:

```
# rpm -i packagename
# rpm -U packagename
# rpm -F packagename
```

The **rpm -i** option installs the package, if it isn't already installed. The **rpm -U** option upgrades any existing package or installs it if an earlier version isn't already installed. The **rpm -F** option upgrades only existing packages. It does not install a package if it wasn't previously installed.

I like to add the **-vh** options with the **rpm** command. These options add verbose mode and use hash marks that can help monitor the progress of the installation. So when I use **rpm** to install a package, I run the following command:

```
# rpm -ivh packagename
```

There's one more thing associated with a properly designed RPM package. When unpacked with the **rpm** command, it checks to see whether it would overwrite any configuration files. The **rpm** command tries to make intelligent decisions about what to do in this situation. As suggested earlier, if the **rpm** command chooses to replace an existing configuration file, it provides a warning (in most cases) similar to:

```
# rpm -i penguin-3.26.i386.rpm
warning: /etc/someconfig.conf saved as /etc/someconfig.conf.rpmsave
```

The **rpm** command normally works in the same fashion when a package is erased with the **-e** switch. If a configuration file has been changed, it's also saved with a .rpmsave extension in the same directory.

Copyright © 2011 by The McGraw-Hill Companies

It's up to you to look at both files and determine what modifications, if any, need to be made. Of course, as not every RPM package is perfect, there's always a risk that such an update would overwrite that critical customized configuration file. In that case, backups are important.

In general, the **rpm** commands to upgrade a package work only if the package being installed is of a newer version. Sometimes, an older version of a package is desirable. As long as there are no security issues with the older package, more administrators may be familiar with slightly older releases. Bugs that may be a problem on a newer package may not exist in an older version of that package. So if you want to "downgrade" a package with the **rpm -i**, **-U**, or **-F** commands, the **--force** switch can help.

If you've already customized a package and upgraded it with* rpm*, go to the saved configuration file. Use it as a guide to change the settings in the new configuration file. But remember, with upgrades, there may be additional required changes. Therefore, you should test the result for every conceivable production environment.

Uninstall an RPM Package

The **rpm -e** command uninstalls a package. But first, RPM checks a few things. It performs a dependency check to make sure no other packages need what you're trying to uninstall. If dependent packages are found, **rpm -e** fails with an error message identifying these packages.

With properly configured RPMs, if you have modified related configuration files, RPM makes a copy of the file, adds an .rpmsave extension to the end of the filename, and then erases the original. It can then proceed with the uninstallation. When the process is complete, it removes the package name from the database.

Be very careful about which packages you remove from a system. Like many other Linux utilities, RPM may silently let you shoot yourself in the foot. For example, if you were to remove the packages that include /etc/passwd or the kernel, it could render that system unusable.

Install RPMs from Remote Systems

With the RPM system, you can even specify package locations similar to an Internet address, in URL format. For example, if you want to apply the **rpm** command to the

Copyright © 2011 by The McGraw-Hill Companies

foo.rpm package on the /pub directory of the ftp.rpmdownloads.com FTP server, you can install this package with a command such as:

```
# rpm -ivh ftp://ftp.rpmdownloads.com/pub/foo.rpm
```

Assuming you have a network connection to that remote server, this particular **rpm** command logs on to the FTP server anonymously and downloads the file. Unfortunately, an attempt to use wildcards in the package name with this command leads to an error message associated with "file not found." The complete package name is required, which can be an annoyance.

If you installed RHEL 6 from an FTP server as instructed in Chapters 1 and 2, you could substitute the associated URL, along with the exact name of the package. For example, based on the FTP server configured in Chapter 1 and the aforementioned policycoreutils-gui package, the appropriate command would be:

```
# rpm -ivh ftp://192.168.122.1/pub/inst/policycoreutils-gui-2.0.83-19.1.el6.
x86_64.rpm
```

If the FTP server requires a username and password, you can include them in the following format: ftp://*username*:*password*@*hostname*:*port*/*path*/*to*/*remote*/*package*/*file*.rpm, where *username* and *password* are the username and password you need to log on to this system, and *port*, if required, specifies a nonstandard port used on the remote FTP server. One of the drawbacks of FTP is that all information is transmitted in clear text. When that information includes usernames and passwords, that can be a security risk. For such reasons, anonymous FTP servers may be preferred.

Based on the preceding example, if the username is *mjang* and the password is *Ila451MS*, you could install an RPM directly from a server with the following command:

```
# rpm -ivh ftp://mjang:Ila451MS@192.168.122.1/pub/inst/policycoreutils-gui-
2.0.83-19.1.el6.x86_64.rpm
```

The key to this system is the **rpm** command. Unfortunately, globbing no longer works with this command, at least when tested on RHEL 6. So the exact name of the package to be installed is required.

RPM Installation Security

Security can be a concern, especially with RPM packages downloaded over the Internet. If a cracker were to somehow penetrate the RHN, or perhaps a third-party

Copyright © 2011 by The McGraw-Hill Companies

repository, how would you know that packages from those sources were genuine? The key is the GNU Privacy Guard (GPG) key, which is the open-source implementation of Pretty Good Privacy (PGP).

If you haven't imported or installed a GPG key, you might have noticed something similar to the following message when packages are installed:

```
warning: vsftpd-2.2.2-6.el6.i686.rpm: Header V3 RSA/SHA256
Signature, key ID f21541eb: NOKEY
```

If you're concerned about security, this warning should raise alarm bells. During the RHEL 6 installation process, GPG keys are stored in the /etc/pki/rpm-gpg directory. Take a look at the contents of this directory. You'll find files like RPM-GPG-KEY-redhat-release. To actually use the key to verify packages, it has to be imported. And the command to import the GPG key is fairly simple:

```
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

If there's no output, the **rpm** command probably successfully imported the GPG key. Even if this command succeeds, if you repeat it, an "import failed" message will appear. In addition, the GPG key is now included in the RPM database, which can be verified with the **rpm -qa gpg-pubkey** command.

In the /etc/pki/rpm-gpg directory, there are normally five GPG keys available, as described in Table 7-1.

Later in this chapter, you'll see how GPG keys are imported automatically from remote repositories when new packages are installed.

TABLE 7-1

rpm --query **Options**

GPG Key	Description
RPM-GPG-KEY-redhat-beta	Packages built for the RHEL 6 beta
RPM-GPG-KEY-redhat-legacy-former	Packages for pre–November 2006 releases (and updates)
RPM-GPG-KEY-redhat-legacy-release	Packages for post–November 2006 releases
RPM-GPG-KEY-redhat-legacy-rhx	Packages associated with Red Hat Exchange
RPM-GPG-KEY-redhat-release	Released packages for RHEL 6

Copyright © 2011 by The McGraw-Hill Companies

Special RPM Procedures with the Kernel

Kernel updates incorporate new features, address security issues, and generally help Linux systems work better. However, kernel updates can also be a pain in the rear end, especially if specialized packages that depend on an existing version of a kernel have been installed.

In any case, don't upgrade a kernel if you're not ready to repeat every step taken to customize software with the existing kernel, whether it's specialized drivers, recompiling to incorporate additional filesystems, or more. For example, the drivers for a few wireless network cards and printers may be tied to a specific version of a kernel. Some virtual machine software components (not including KVM) may be installed against a specific version of a kernel. In those cases, upgrading a kernel may mean that you'd also have to rebuild associated drivers to keep those network cards, printers, and virtual machine systems operational.

If you see an available update for a kernel RPM, the temptation is to run the **rpm -U *newkernel*** command. Don't do it! It overwrites your existing kernel, and if the updated kernel doesn't work with the system, you're out of luck. (Well, not completely out of luck, but if you reboot and have problems, you'll have to use **linux rescue** mode discussed in Chapter 5 to boot a system and reinstall the existing kernel. In the days where there was a separate Troubleshooting and System Maintenance section on the Red Hat exams, that might have made for an interesting test scenario.)

The best option for upgrading to a new kernel is to install it, specifically with a command such as:

```
# rpm -ivh newkernel
```

If you're connected to an appropriate repository, the following command works just as well:

```
# yum install kernel
```

This installs the new kernel, along with related files, side by side with the current working kernel. One example of the result is shown in Figure 7-1, in the output to the **ls /boot** command.

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-1

New and existing kernel files in the /boot directory

```
[root@server1 ~]# ls /boot/
config-2.6.32-71.7.1.el6.x86_64            lost+found
config-2.6.32-71.el6.x86_64                symvers-2.6.32-71.7.1.el6.x86_64.gz
efi                                        symvers-2.6.32-71.el6.x86_64.gz
grub                                       System.map-2.6.32-71.7.1.el6.x86_64
initramfs-2.6.32-71.7.1.el6.x86_64.img     System.map-2.6.32-71.el6.x86_64
initramfs-2.6.32-71.el6.x86_64.img         vmlinuz-2.6.32-71.7.1.el6.x86_64
initrd-2.6.32-71.7.1.el6.x86_64kdump.img   vmlinuz-2.6.32-71.el6.x86_64
[root@server1 ~]#
```

You'll note different files for various parts of the boot process in the boot directory, briefly described in Table 7-2. In many cases, you won't see a new initrd file with a new kernel, unless the system has sufficient RAM (several GB) to support kernel crash dumps. On my system, RHEL 6 did not support crash dumps until I upgraded from 4GB to 8GB of RAM. So unless you have more than 4GB of RAM, don't be concerned if you don't see a new initrd-* file when a new kernel is installed.

The installation of a new kernel adds options to boot the new kernel in the boot loader menu, as defined in the GRUB configuration file (/boot/grub/grub.conf), without erasing existing options. One example of the revised GRUB configuration file is shown in Figure 7-2.

TABLE 7-2

Files in the /boot Directory

File	Description
config-*	Kernel configuration settings; a text file
efi/	Extensible firmware interface (EFI) directory; includes an interface between GRUB and the newer UEFI
grub/	Directory with GRUB configuration files
initramfs-*	The initial RAM disk filesystem, a root filesystem called during the boot process to help load other components
initrd-*	A root filesystem used for kernel crash dumps
symvers-*	List of modules
System.map-*	Map of system names for variables and functions, with their locations in memory
vmlinuz	The actual Linux kernel

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-2

GRUB with a second kernel

```
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/vda2
#          initrd /initrd-[generic-]version.img
#boot=/dev/vda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-71.7.1.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-71.7.1.el6.x86_64 ro root=UUID=d92cce5c-d3da-40f
8-9924-6ebe137560a2 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYS
FONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=auto rhgb quiet
        initrd /initramfs-2.6.32-71.7.1.el6.x86_64.img
title Red Hat Enterprise Linux (2.6.32-71.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=UUID=d92cce5c-d3da-40f8-99
24-6ebe137560a2 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT
=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=auto rhgb quiet
        initrd /initramfs-2.6.32-71.el6.x86_64.img
~
~
```

A careful reading of the two stanzas reveals that the only difference is in the version numbers listed in the title, for the Linux kernel, and for the Initial RAM disk filesystem. In addition, the default kernel is the newly installed kernel. So if that kernel does not work, you can restart the system, access the GRUB menu, and then boot from the older kernel, which presumably still works.

If for some reason an updated kernel does not update the GRUB bootloader, all you need to do is make a copy of one stanza, and revise the version number for the title, the kernel, and the Initial RAM disk filesystem filenames.

CERTIFICATION OBJECTIVE 7.02

More RPM Commands

The **rpm** command is rich with details. All this book can do is cover some of the basic ways **rpm** can help you manage RHEL. You've already read about how **rpm** can install and upgrade packages in various ways. Queries can help you identify what's installed, in detail. Validation tools can help you check the integrity of packages and individual files. You can use related tools to help identify the purpose of different RPMs, as well as a full list of those RPMs already installed.

Copyright © 2011 by The McGraw-Hill Companies

The RHCE objectives include a requirement to create a simple RPM package. The building of source code, along with related commands, is discussed in Chapter 12.

Package Queries

The simplest RPM query verifies whether a specific package is installed. The following command verifies the installation of the Upstart package (the version number you see may vary):

```
# rpm -q upstart
upstart-0.6.5-6.1.el6_0.1.x86_64
```

You can do more with RPM queries, as described in Table 7-3. Note how queries are associated with **-q** or **--query**; full-word switches such as **--query** are usually associated with a double-dash.

Package Signatures

RPM uses several methods for checking the integrity of a package. You've seen how to import the GPG signature. Some of these methods are shown in the output to the **rpm --checksig *pkgname*** command. (The **-K** switch is equivalent to **--checksig**.) For example, if you've downloaded a package from a third party such as the hypothetical pkg-1.2.3-4.noarch.rpm package, and want to check it against the imported GPG signature, run the following command:

```
# rpm --checksig pkg-1.2.3-4.noarch.rpm
```

If successful, you'll see output similar to the following:

```
pkg-1.2.3-4.noarch.rpm: rsa sha1 (md5) pgp md5 OK
```

TABLE 7-3 rpm --query **Options**

rpm Query Command	Meaning
rpm -qa	Lists all installed packages.
rpm -qf /path/to/file	Identifies the package associated with /path/to/file.
rpm -qc ***packagename***	Lists only configuration files from *packagename*.
rpm -qi ***packagename***	Displays basic information for *packagename*.
rpm -ql ***packagename***	Lists all files from *packagename*.
rpm -qR ***packagename***	Notes all dependencies; you can't install *packagename* without them.

Copyright © 2011 by The McGraw-Hill Companies

You may already recognize the algorithms used to verify package integrity:

- **rsa** Named for its creators, Rivest, Shamir, and Adlemen, it's a public key encryption algorithm.
- **sha1** A 160-bit message digest Secure Hash Algorithm; a cryptographic hash function.
- **md5** Message Digest 5, a cryptographic hash function.
- **pgp** PGP, as implemented in Linux by GPG.

File Verification

The verification of an installed package compares information about that package with information from the RPM database on a system. The **--verify** (or **-v**) switch checks the size, MD5 checksum, permissions, type, owner, and group of each file in the package. Verification can be done in a number of ways. Here are a few examples:

- Verify all files. Naturally, this may take a long time on your system. (Of course, the **rpm -Va** command performs the same function.)

```
# rpm --verify –a
```

- Verify all files within a package against a downloaded RPM.

```
# rpm --verify -p vsftpd-2.2.2-6.el6.i686.rpm
```

- Verify a file associated with a particular package.

```
# rpm --verify --file /bin/ls
```

If the integrity of the files or packages is verified, you will see no output. Any output means that a file or package is different from the original. There's no need to panic if certain changes appear; after all, administrators do edit configuration files. There are eight tests. If there's been a change, the output is a string of up to eight failure code characters, each of which tells you what happened during each test.

If you see a dot (.), that test passed. The following example shows /bin/vi with an incorrect group ID assignment:

```
# rpm --verify --file /bin/vi
......G.   /bin/vi
```

Table 7-4 lists the failure codes and their meanings.

Copyright © 2011 by The McGraw-Hill Companies

TABLE 7-4

rpm --verify **Codes**

Failure Code	Meaning
5	MD5 checksum
S	File size
L	Symbolic link
T	File modification time
D	Device
U	User
G	Group
M	Mode

Now here's an interesting experiment: When you have one version of a package installed, use the **rpm --verify -p** command with a second version of the same package. Finding such a package should not be too difficult, as Red Hat updates packages for feature updates, security patches, and yes, bug fixes, frequently. For example, when I wrote this book for RHEL 6, I had access to both vsftpd-2.2.2-1.el6.i686.rpm and vsftpd-2.2.2-6.el6.i686.rpm When the latter version was installed, I ran the following command:

```
# rpm --verify -p vsftpd-2.2.2-1.el6.i686.rpm
```

and got a whole list of changed files, as shown in Figure 7-3. That command provides information on what was changed between different versions of the vsFTP server package.

FIGURE 7-3

Verifying changes between packages

```
[root@server1 ~]# rpm --verify -p rpmbuild/RPMS/i686/vsftpd-2.2.2-1.el6.i686.rpm

.......T.  c /etc/logrotate.d/vsftpd
.......T.  c /etc/pam.d/vsftpd
S.5....T.    /etc/rc.d/init.d/vsftpd
.......T.  c /etc/vsftpd/ftpusers
.......T.  c /etc/vsftpd/user_list
S.5....T.  c /etc/vsftpd/vsftpd.conf
.......T.    /etc/vsftpd/vsftpd_conf_migrate.sh
..5....T.    /usr/sbin/vsftpd
.......T.  d /usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/README
.......T.  d /usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE_NOINETD/README
.......T.  d /usr/share/doc/vsftpd-2.2.2/EXAMPLE/PER_IP_CONFIG/README
.......T.  d /usr/share/doc/vsftpd-2.2.2/EXAMPLE/VIRTUAL_USERS/README
.......T.  d /usr/share/doc/vsftpd-2.2.2/FAQ
.......T.  d /usr/share/doc/vsftpd-2.2.2/INSTALL
.......T.  d /usr/share/doc/vsftpd-2.2.2/README
.......T.  d /usr/share/doc/vsftpd-2.2.2/vsftpd.xinetd
S.5....T.  d /usr/share/man/man5/vsftpd.conf.5.gz
.......T.  d /usr/share/man/man8/vsftpd.8.gz
[root@server1 ~]#
```

Copyright © 2011 by The McGraw-Hill Companies

Different Databases of Installed Packages

There are two basic databases of installed RPMs. Through RHEL 5, the basic list was stored in /var/log/rpmpkgs. However, as that database was updated only once a day, it's often best to just get the current list of installed packages with the following command:

```
# rpm -qa
```

The /root/install.log file includes all packages included when RHEL was installed on this system. It's not updated after installation. So once a new or updated package is installed, that file is out of date. If desired, you can set up that same /var/log/rpmpkgs database in RHEL 6, by installing the rpm-cron package.

CERTIFICATION OBJECTIVE 7.03

Dependencies and the yum Command

The **yum** command makes it easy to add and remove software packages to a system. It maintains a database regarding the proper way to add, upgrade, and remove packages. This makes it relatively simple to add and remove software with a single command. That single command overcame what was known as "dependency hell."

The **yum** command was originally developed for Yellow Dog Linux. The name is based on the Yellow Dog updater, modified. Given the trouble associated with dependency hell, Linux users were motivated to find a solution. It was adapted for Red Hat distributions with the help of developers from Duke University.

The configuration of **yum** depends on package libraries known as repositories. Red Hat repositories are configured through the RHN. As the repositories of third-party rebuild distributions can't use RHN (without a subscription), they use publicly available servers. In either case, it's important to know the workings of the **yum** command, how it installs and updates individual packages as well as package groups.

An Example of Dependency Hell

To understand more about the need for the **yum** command, examine Figure 7-4. The packages listed in that figure are what's required to build an RPM. While the building of an RPM package is a RHCE requirement, the associated packages provide an excellent illustration of the need for **yum**.

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-4

Packages required to build RPMs

```
[root@server1 SPECS]# rpmbuild -bp --target=`uname -m` kernel.spec
Building target platforms: x86_64
Building for target x86_64
error: Failed build dependencies:
        gcc >= 3.4.2 is needed by kernel-2.6.32-71.7.1.el6.x86_64
        redhat-rpm-config is needed by kernel-2.6.32-71.7.1.el6.x86_64
        patchutils is needed by kernel-2.6.32-71.7.1.el6.x86_64
        elfutils-libelf-devel is needed by kernel-2.6.32-71.7.1.el6.x86_64
        zlib-devel is needed by kernel-2.6.32-71.7.1.el6.x86_64
        binutils-devel is needed by kernel-2.6.32-71.7.1.el6.x86_64
        hmaccalc is needed by kernel-2.6.32-71.7.1.el6.x86_64
[root@server1 SPECS]#
```

You could try to use the **rpm** command to install each of these packages. To do so, take the following steps:

1. Include the RHEL 6 DVD. Insert it into its drive, or make sure it's included in the configuration for the target virtual machine.
2. Unless it's already mounted, mount that DVD with the following command. Of course, a different empty directory can be substituted for /media.

   ```
   # mount /dev/sr0 /media
   ```

3. Navigate to the directory where the DVD is mounted, /media or some subdirectory of /media.
4. The RPM packages on the RHEL 6 DVD can be found in the Packages/ subdirectory of the DVD. Navigate to that subdirectory.
5. Enter the **rpm -ivh** command, and then type in the names of the packages listed in Figure 7-4. It may be easiest to use command completion for this purpose; for example, if you were to type in:

   ```
   # rpm -ivh gcc-
   ```

You could then press the TAB key twice, and review available packages that start with *gcc-*. You could then enter additional keys and press the TAB key again to complete the name of the package. After a bit of work, you'd end up with something similar to the command and results shown in Figure 7-5. What actually appears depends on the current revision level of each package, as well as what's already installed on the local system.

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-5

These packages have dependencies.

```
[root@tester1 Packages]# rpm -ivh gcc-4.4.4-13.el6.x86_64.rpm redhat-rpm-config-9.0.3-25.el6.
noarch.rpm patchutils-0.3.1-3.1.el6.x86_64.rpm elfutils-libelf-devel-0.148-1.el6.x86_64.rpm z
lib-devel-1.2.3-25.el6.x86_64.rpm binutils-devel-2.20.51.0.2-5.11.el6.x86_64.rpm hmaccalc-0.9
.12-1.el6.x86_64.rpm
warning: gcc-4.4.4-13.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
error: Failed dependencies:
        cloog-ppl >= 0.15 is needed by gcc-4.4.4-13.el6.x86_64
        cpp = 4.4.4-13.el6 is needed by gcc-4.4.4-13.el6.x86_64
        glibc-devel >= 2.2.90-12 is needed by gcc-4.4.4-13.el6.x86_64
[root@tester1 Packages]#
```

6. The next step is to try to include these dependencies in the list of packages to be installed. When I try this step, it leads to more dependencies, as shown in Figure 7-6.

At this point, the addition of more packages to the installation becomes somewhat more difficult. How would you know, except from experience, that the ppl-* and mpfr-* packages would satisfy the first three "Failed Dependencies" error messages? Even if you do already understand, the inclusion of such packages is not enough. There's even one more level of dependent packages. This pain is known as dependency hell.

Relief from Dependency Hell

Before **yum**, some attempts to use the **rpm** command were stopped by the dependencies described earlier. Sure, you could install those dependent packages with the same command, but what if those dependencies themselves have dependencies? That perhaps is the biggest advantage of the **yum** command.

Before **yum**, RHEL incorporated dependency resolution into the update process. Through RHEL 4, this was done with **up2date**. Red Hat incorporated **yum** starting

FIGURE 7-6

There are even more dependencies.

```
[root@tester1 Packages]# rpm -ivh gcc-4.4.4-13.el6.x86_64.rpm redhat-rpm-config-9.0.3-25.el6.
noarch.rpm patchutils-0.3.1-3.1.el6.x86_64.rpm elfutils-libelf-devel-0.148-1.el6.x86_64.rpm z
lib-devel-1.2.3-25.el6.x86_64.rpm binutils-devel-2.20.51.0.2-5.11.el6.x86_64.rpm hmaccalc-0.9
.12-1.el6.x86_64.rpm cloog-ppl-0.15.7-1.2.el6.x86_64.rpm cpp-4.4.4-13.el6.x86_64.rpm glibc-de
vel-2.12-1.7.el6.x86_64.rpm
warning: gcc-4.4.4-13.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
error: Failed dependencies:
        libppl.so.7()(64bit) is needed by cloog-ppl-0.15.7-1.2.el6.x86_64
        libppl_c.so.2()(64bit) is needed by cloog-ppl-0.15.7-1.2.el6.x86_64
        libmpfr.so.1()(64bit) is needed by cpp-4.4.4-13.el6.x86_64
        glibc-headers is needed by glibc-devel-2.12-1.7.el6.x86_64
        glibc-headers = 2.12-1.7.el6 is needed by glibc-devel-2.12-1.7.el6.x86_64
[root@tester1 Packages]#
```

Copyright © 2011 by The McGraw-Hill Companies

with RHEL 5. The **yum** command uses subscribed RHN channels and any other repositories configured in the /etc/yum.repos.d directory.

All you need to do to install the packages listed in Figure 7-4 is run the following command:

```
# yum install gcc redhat-rpm-config patchutils elfutils-libelf-devel zlib-devel
binutils-devel hmaccalc
```

If so prompted, accept the request to install additional dependent packages, and then all of the noted dependencies are installed automatically. (Yes, the **-y** switch would perform the same function.) If updates are available from connected repositories, the latest available version of each package is installed. The **yum** command is described in more detail later in this chapter.

But if you're running RHEL 6 without a connection to the RHN, nothing happens. Shortly, you'll see how to create a connection between **yum** and the installation server created in Chapter 1.

There are a number of third-party repositories available for RHEL. They include several popular applications that are not supported by Red Hat. For example, I use one to install packages associated with my laptop wireless network card.

While the owners of these repositories work closely with some Red Hat developers, there are some reports where dependencies required from one repository are unavailable from other repositories, leading to a different form of "dependency hell." However, at least the more popular third-party repositories are excellent; I've never encountered "dependency hell" from using these repositories.

There are two main reasons why Red Hat does not include most proven and popular packages available from third-party repositories. Some are not released under open-source licenses, and others are packages that Red Hat simply chooses not to support.

Basic yum Configuration

Relief from dependency hell depends on the proper configuration of **yum**. Not only do you need to know how to configure **yum** to connect to repositories over the Internet, but also you need to know how to configure **yum** to connect to repositories on a local network. With this knowledge, you can connect **yum** to repositories on the RHN, to repositories configured by third parties, and to custom repositories configured for specialized networks. And remember, during the Red Hat exams, you won't have access to the Internet.

Copyright © 2011 by The McGraw-Hill Companies

To that end, you to understand how yum is configured in some detail. It starts with the /etc/yum.conf configuration file and continues with files in the /etc/yum and /etc/yum.repos.d directories. To get the full list of yum configuration directives and their current values, run the following command:

```
# yum-config-manager
```

This command requires the installation of the yum-utils package.

The Basic yum Configuration File: yum.conf

This section analyzes the default version of the /etc/yum.conf file, line by line. While you won't make changes to this file in most cases, you need to understand at least the standard directives in this file, if something goes wrong. The following lines are straight excerpts from the default version of this file. The first directive is a header; the **[main]** header suggests that all directives that follow apply globally to **yum**.

```
[main]
```

The **cachedir** directive specifies where caches of packages, package lists, and related databases are to be downloaded. Based on the standard 64-bit architecture for RHEL 6, this translates to the /var/cache/yum/x86_64/6Server directory.

```
cachedir=/var/cache/yum/$basearch/$releasever
```

The **keepcache** boolean directive specifies whether **yum** actually stores downloaded headers and packages in the directory specified by **cachedir**. The standard shown here suggests that caches are not kept, which helps make sure that a system is kept up to date with the latest available packages.

```
keepcache=0
```

The **debuglevel** directive is closely related to the **errorlevel** and **logfile** directives, as they specify the detail associated with debug and error messages. Even though the **errorlevel** directive is not shown, both it and **debuglevel** are set to 2 by default. The available range is 0–10, where 0 provides almost no information, and 10 provides perhaps too much information even for developers.

```
debuglevel=2
logfile=/var/log/yum.log
```

Copyright © 2011 by The McGraw-Hill Companies

The **exactarch** boolean directive makes sure the architecture matches the actual processor type, as defined by the **arch** command.

```
exactarch=1
```

The **obsoletes** boolean directive can support the uninstallation of obsolete packages in conjunction with a **yum update** command.

```
obsoletes=1
```

The **gpgcheck** boolean directive makes sure the **yum** command actually checks the GPG signature of downloaded packages.

```
gpgcheck=1
```

The **plugins** boolean directive provides a necessary link to Python-based RHN plugins in the /usr/share/yum-plugins directory. It also refers indirectly to plugin configuration files in the /etc/yum/pluginconf.d directory.

```
plugins=1
```

The **installonly_limit** directive provides a safeguard of sorts, making sure that Linux kernel packages are always installed and not upgraded, for reasons described earlier in this chapter:

```
installonly_limit=3
```

To make sure the header data downloaded from the RHN (and any other repositories) are up to date, the **metadata_expire** directive specifies a lifetime for headers. The default is shown in comments. In other words, if you haven't used the **yum** command in 90 minutes, the next use of the **yum** command downloads the latest header information.

```
#metadata_expire=90m
```

The final directive of interest, in comments, happens to be the default; it's a reference to the noted directory for actual configuration information for repositories:

```
# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

Copyright © 2011 by The McGraw-Hill Companies

Configuration Files in the /etc/yum/pluginconf.d Directory

The default files in the /etc/yum/pluginconf.d directory configure a connection between **yum** and the Red Hat network. If you're studying from a RHEL rebuild distribution such as Scientific Linux, you'll see a different set of files in this directory. In Scientific Linux, the files in this directory are focused on connecting the local system to better repositories over the Internet. But as this is a Red Hat book, the focus will be on the two basic files in the RHEL 6 installation.

If you've installed the Kickstart Configurator discussed in Chapter 2, there will be two additional files in this directory: blacklist.conf and whiteout.conf. They are beyond the scope of the RHCSA and RHCE exams.

Red Hat Network Plugins with rhnplugin.conf

If you have a subscription to the RHN, the rhnplugin.conf file in this directory is especially important. While the directives, as follows, may seem simple, they enable access and check GPG signatures:

```
[main]
enabled = 1
gpgcheck = 1
```

In comments, this file suggests that different settings can be configured for different repositories. The repositories entitled in brackets should match those associated with the actual RHN repositories (or the repositories of third-party rebuilds).

Red Hat Network Plugins with refresh-packagekit.conf

The refresh-packagekit-conf file is designed to connect the **yum** system to the PackageKit. As discussed later in this chapter, PackageKit is a system designed to work with all types of update commands, including **yum**, **apt**, and others. It's a simple file, with two directives, which enables a connection between **yum** and the GUI package management tools discussed later in this chapter.

```
[main]
enabled=1
```

Copyright © 2011 by The McGraw-Hill Companies

Configuration Files in the /etc/yum.repos.d Directory

The configuration files in the /etc/yum.repos.d directory are designed to connect systems to actual repositories. If you're running a rebuild distribution such as Scientific Linux, you'll see files that connect to public repositories on the Internet. If you're running RHEL 6, this directory may be empty, unless that system was installed locally from the RHEL 6 DVD. In that case, you'll see a packagekit-media.repo file in that directory, which is designed to get further updates from the DVD.

A couple of elements in common for configuration files in the /etc/yum.repos.d directory is the file extension (.repo) and the documentation, available with the **man yum.conf** command.

A properly configured .repo file in the /etc/yum.repos.d directory can be a terrific convenience, to enable the installation of groups of packages with the **yum** command. As the /etc/yum.repos.d directory may be empty on a RHEL 6 system, you should know how to create that file from scratch, using data for the installation server and information available in the yum.conf man page.

Deal with the packagekit-media.repo File

However, as the latest updates are available, it is often best to disable that file. In fact, you may not even have the RHEL 6 installation DVD available during an exam. While you could just delete that file, other software components regenerate that file when RHEL 6 is rebooted. So the best approach to the packagekit-media.repo file is to disable it by adding the following directive to the end of the file:

```
enabled=0
```

If you're experienced with various versions of Fedora Linux, this solution hasn't always worked. However, it works in current tests of RHEL 6 systems installed from DVD (as opposed to the RHCSA objective of installing RHEL 6 over a network).

Understand /etc/yum.repos.d Configuration Files for Rebuild Distributions

If you're running a rebuild distribution, the files in the /etc/yum.repos.d directory may connect the local system to one or more remote repositories. One example comes from Scientific Linux 6, as shown in Figure 7-7. While it includes a number of different repositories, you can learn from the pattern of directives configured for each repository.

Copyright © 2011 by The McGraw-Hill Companies

There are four stanzas of data shown in Figure 7-7. Each stanza represents a connection to a Scientific Linux repository. For example, the first stanza includes the basic elements of a repository and more. The first line, in brackets, provides a name for the repository. In this case, the [sl] just happens to match the initials of Scientific Linux. It doesn't represent the directory where the associated packages are installed. That's a difference with the /etc/yum/pluginconf.d/rhnplugin.conf file described earlier.

```
[sl]
```

FIGURE 7-7

Several repositories configured in one file

```
[sl]
name=Scientific Linux $releasever - $basearch
baseurl=http://ftp.scientificlinux.org/linux/scientific/$releasever/$basearch/os/
                http://ftp1.scientificlinux.org/linux/scientific/$releasever/$basearch/os/
                http://ftp2.scientificlinux.org/linux/scientific/$releasever/$basearch/os/
                ftp://ftp.scientificlinux.org/linux/scientific/$releasever/$basearch/os/
#mirrorlist=http://ftp.scientificlinux.org/linux/scientific/mirrorlist/sl-base-6.txt
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-sl file:///etc/pki/rpm-gpg/RPM-GPG-KEY-dawson

[sl-testing]
name=Scientific Linux $releasever Testing - $basearch
baseurl=http://ftp.scientificlinux.org/linux/scientific/6rolling/testing/$basearch/
                http://ftp1.scientificlinux.org/linux/scientific/6rolling/testing/$basearch/
                http://ftp2.scientificlinux.org/linux/scientific/6rolling/testing/$basearch/
                ftp://ftp.scientificlinux.org/linux/scientific/6rolling/testing/$basearch/
enabled=0
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-sl file:///etc/pki/rpm-gpg/RPM-GPG-KEY-dawson

[sl-source]
name=Scientific Linux $releasever Alpha - Source
baseurl=http://ftp.scientificlinux.org/linux/scientific/$releasever/SRPMS/
                http://ftp1.scientificlinux.org/linux/scientific/$releasever/SRPMS/
                http://ftp2.scientificlinux.org/linux/scientific/$releasever/SRPMS/
                ftp://ftp.scientificlinux.org/linux/scientific/$releasever/SRPMS/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-sl file:///etc/pki/rpm-gpg/RPM-GPG-KEY-dawson

[sl-testing-source]
name=Scientific Linux $releasever Testing - Source
baseurl=http://ftp.scientificlinux.org/linux/scientific/6rolling/testing/SRPMS/
                http://ftp1.scientificlinux.org/linux/scientific/6rolling/testing/SRPMS/
                http://ftp2.scientificlinux.org/linux/scientific/6rolling/testing/SRPMS/
                ftp://ftp.scientificlinux.org/linux/scientific/6rolling/testing/SRPMS/
enabled=0
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-sl file:///etc/pki/rpm-gpg/RPM-GPG-KEY-dawson
```

Copyright © 2011 by The McGraw-Hill Companies

However, when you run the **yum update** command to update the local database of those remote packages, it includes **sl** as the name of the repository, in output similar to the following, which suggests that it took ten seconds to download the 1.6MB database of existing repository data.

```
sl            | 1.6MB   00:10
```

While the name of the repository follows, it's just for documentation purposes and does not affect how packages or package databases are read or downloaded. However, the inclusion of the **name** directive does avoid a nonfatal error message.

```
name=Scientific Linux $releasever - $basearch
```

Note the several **baseurl** directives that follow. While only one is required, multiple **baseurl** directives specify the URL to different remote servers with a copy of the actual repository of packages. It can work with either the HTTP or the FTP protocol. (It can even work with local directories or mounted Network File System shares, as described in Exercise 7-1.)

```
baseurl=http://ftp.scientificlinux.org/linux/scientific/$releasever/$basearch/os/
```

Alternatively, these repositories can be set up in a file downloaded with the **mirrorlist** directive:

```
#mirrorlist=http://ftp.scientificlinux.org/linux/scientific/mirrorlist/sl-base-
6.txt
```

While repositories configured in .repo files in the /etc/yum.repos.d directory are **enabled** by default, the following directive provides an easy way to deactivate a connection to such (**enabled=0** would deactivate the connection):

```
enabled=1
```

If you want **yum** to check the GPG signatures of each package to be downloaded, the following command puts that wish into effect:

```
gpgcheck=1
```

Of course, any GPG check requires a GPG key; the following directive specifies two keys from the local /etc/pki/rpm-gpg directory for that purpose:

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-sl file:///etc/pki/rpm-gpg/RPM-GPG-KEY-
dawson
```

Copyright © 2011 by The McGraw-Hill Companies

Create Your Own /etc/yum.repos.d Configuration File

You'll want to know how to create a local configuration file in the /etc/yum.repos.d directory. It enables the use of the **yum** command, which is the easiest way to install groups of packages like the Apache web server from Chapter 1 or any of the groups of packages discussed in the RHCE part of this book.

To do so, you'll need to set up a text file with a .repo extension in the /etc/yum .repos.d directory. All that file needs is three lines. In fact, if you're willing to accept some nonfatal errors, two lines are sufficient.

On RHEL 6, especially during an exam, the /etc/yum.repos.d directory may be empty. So you may not have access to examples such as those available for Scientific Linux, as shown in Figure 7-7. The first guidance comes from the following comments at the bottom of the /etc/yum.conf file, which confirm that the file must have a .repo extension in the /etc/yum.repos.d directory:

```
# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

In addition, you could configure the three lines in the /etc/yum.conf file. If you forget what three lines to add, there is an example in the man page for the yum.conf file, as shown in Figure 7-8.

FIGURE 7-8 Excerpt from man yum.conf for a new /etc/yum .repos.d file

```
[repository] OPTIONS
      The repository section(s) take the following form:

            Example: [repositoryid]
            name=Some name for this repository
            baseurl=url://path/to/repository/

            repositoryid  Must  be  a  unique  name for each repository, one
            word.

            name A human readable string describing the repository.

            baseurl Must be a URL to the directory  where  the  yum  reposi-
            tory's  'repodata' directory lives. Can be an http://, ftp:// or
            file:// URL. You can specify multiple URLs in one baseurl state-
            ment. The best way to do this is like this:
            [repositoryid]
            name=Some name for this repository
            baseurl=url://server1/path/to/repository/
                    url://server2/path/to/repository/
                    url://server3/path/to/repository/

            If you list more than one baseurl= statement in a repository you
            will find yum will ignore the  earlier  ones  and  probably  act
            bizarrely. Don't do this, you've been warned.

:
```

Copyright © 2011 by The McGraw-Hill Companies

If you forget what to do, run the **man yum.conf** command, and scroll down to this part of the man page. The identifier for the repository is shown in brackets. Unless specified by the RHCSA exam, it doesn't matter what single word you put between the brackets as the identifier.

For the purpose of this chapter, I open a new file named whatever.repo in the /etc/yum.repos.d directory. (To some extent, the filename of the .repo file does not matter, as long as it has a .repo extension in the /etc/yum.repos.d directory.) In that file, I add the following identifier:

```
[test]
```

exam Watch

While not required, you should know how to create a working .repo file in the /etc/yum.repos.d directory during the RHCSA and RHCE exams. It can be a big time saver when you need to install additional packages.

Next comes the **name** directive for the repository. As suggested by the listing in the man page, that name should be "human readable." In Linux parlance, that also means the name does not affect the functionality of the repository. To demonstrate, I add the following directive:

```
name=somebody likes Linux
```

Finally, there's the **baseurl** directive, which can be configured to point to an installation server. Per the RHCSA requirements, you need to know how to install Linux from a remote server. It also suggests that you need to know how to install and update packages from a remote repository. To meet either objective, you need to know the URL of that remote server or repository. It's reasonable to expect that URL to be provided during the exam. In Chapter 1, you created FTP and HTTP installation servers on the host system for virtual machines, which are "remote" from those systems.

The FTP and HTTP installation servers that you created in Chapter 1 can also be used as remote repositories. To set up access to those repositories, all you need to include is one of the following **baseurl** directives:

```
baseurl=ftp://192.168.122.1/pub/inst
baseurl=http://192.168.122.1/inst
```

As suggested in the yum.conf man page, you should not include both URLs in separate **baseurl** directives. Make a choice and save the resulting file. That's all you need. There's no reason (except for better security) to include the **enabled**, **gpgcheck**, or **gpgkey** directives described earlier. Of course security is important in real life, but if your focus is on the exam, the best advice is often to keep things as simple as possible.

Copyright © 2011 by The McGraw-Hill Companies

Once the file is saved, run the following commands, to first clear out databases from previously accessed repositories, and then to update the local database from the repository newly configured in the /etc/yum.repos.d/whatever.repo file.

```
# yum clean all
# yum update
```

For a system not registered with the RHN, it'll lead to the following output:

```
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
test                                          | 3.7 kB     00:00
test/primary_db                               | 2.9 MB     00:00
Setting up Update Process
No Packages marked for Update
```

The system is now ready for the installation of new packages. Try running the following command:

```
# yum install system-config-printer
```

Given the virtual machines configured earlier in this book, you might see the result shown in Figure 7-9. If confirmed, the **yum** command would download and then install not only the system-config-printer RPM, but also the four dependent packages shown in the figure to make sure the system-config-printer package is fully supported.

FIGURE 7-9

Installation of one package can include dependencies.

```
Dependencies Resolved

================================================================================
 Package                    Arch          Version              Repository    Size
================================================================================
Installing:
 system-config-printer      x86_64        1.1.16-17.el6        test         444 k
Installing for dependencies:
 gnome-python2-gnomekeyring x86_64        2.28.0-4.el6         test          24 k
 libsmbclient               x86_64        3.5.4-68.el6         test         1.7 M
 notify-python              x86_64        0.1.1-10.el6         test          26 k
 system-config-printer-libs x86_64        1.1.16-17.el6        test         651 k

Transaction Summary
================================================================================
Install       5 Package(s)
Upgrade       0 Package(s)

Total download size: 2.8 M
Installed size: 12 M
Is this ok [y/N]:
```

Copyright © 2011 by The McGraw-Hill Companies

EXERCISE 7-1

Create a yum Repository from the RHEL 6 DVD

This exercise requires access to the RHEL 6 DVD. If you don't have a lot of space for this exercise, it's acceptable to set up the repository directly on the mounted DVD. Alternatively, you can copy the contents to a specified directory. It also assumes an available installation repository, such as one of those created in Chapter 1.

This exercise assumes you'll be starting with no files in the /etc/yum.repos.d directory described in this chapter.

1. If there are existing files in the /etc/yum.repos.d directory, copy them to a backup location such as the root user's home directory, /root.
2. Mount the RHEL 6 DVD on the /mnt directory with the following command (you may need to substitute /dev/cdrom or /dev/dvd for /dev/sr0):

   ```
   # mount /dev/sr0 /mnt
   ```

   Alternatively, if you have only the RHEL 6 DVD as an ISO file, mount it with the following command:

   ```
   # mount -o loop rhel-server-6.0-i386-dvd.iso /mnt
   ```

   Of course, if desired, you can copy the files from a different mount point such as /media to the /mnt directory with a command like **cp -ar /media/. /mnt**. The dot (.) in front of the /media directory ensures the copying of hidden files from the mounted DVD.
3. Open a new file in a text editor. Use a name like rhel6.repo.
4. Edit the rhel6.repo file. Create a new stanza of directives. Use an appropriate stanza title such as **[rhel]**.
5. Specify an appropriate **name** directive for the repository.
6. Include a **baseurl** directive set to **file:///mnt/**. Include an **enabled=1** directive.
7. Save and close the file.
8. Assuming you're running RHEL 6 (and not a rebuild distribution), open the rhnplugin.conf file in the /etc/yum/pluginconf.d directory, and set **enabled=0**.

Copyright © 2011 by The McGraw-Hill Companies

9. Run the **yum clean all** and **yum update** commands.
10. If successful, you should see the following output:

```
Loaded plugins: refresh-packagekit

rhel      | 3.7 kB     00:00 ...
rhel/primary_db      | 2.3 MB     00:00 ...
Setting up Update Process
No Packages marked for Update
:
```

    You've now set up a repository on the local /mnt directory.

11. Restore the original files. Open the rhnplugin.conf file in the /etc/yum/pluginconf.d directory, and set **enabled=1**. Restore the files backed up to the /root directory to the /etc/yum.repos.d directory. If you want to restore the original configuration, delete or move the rhel6.repo file from that directory. Run the **yum clean all command** again.

Third-Party Repositories

Other groups of third-party developers create packages for RHEL 6. They include packages for some popular software not supported by Red Hat. The web sites for two of these third parties can be found at https://rpmrepo.org/RPMforge and http://atrpms.net.

To add third-party repositories to a system, you'd create a custom .repo file in the /etc/yum.repos.d directory. For example, I often use Axel Thimm's third-party repository for my RHEL and Fedora Core systems. It's available from http://ATrpms.net. To make it work with my RHEL system, I use the instructions available from that web site and add the following information to atrpms.repo in the /etc/yum.repos.d directory:

```
[atrpms]
name=atrpms for RHEL $releasever - $basearch
baseurl=http://dl.atrpms.net/el6-i386/atrpms/stable
gpgkey=http://atrpms.net/RPM-GPG-KEY.atrpms
gpgcheck=1
```

If you want to disable any repository in the /etc/yum.repos.d directory, add the following directive to the applicable repository file:

```
enabled=0
```

Copyright © 2011 by The McGraw-Hill Companies

Basic yum Commands

If you want to learn more about the intricacies of the **yum** command, run the command by itself. You'll see the following output scroll by, probably far too fast. Of course, you can pipe the output to the **less** command pager with the **yum | less** command.

```
# yum
Loaded plugins: refresh-packagekit, rhnplugin
You need to give some command
usage: yum [options] COMMAND

List of Commands
```

You'll examine how a few of these commands and options work in the following sections. While you won't have Internet access during a Red Hat exam, you might have a network connection to a locally configured repository. Such configurations are even supported by a variation of the RHN known as the RHN Satellite Server. So you should be ready to configure an appropriate file in the /etc/yum.repos.d directory, as described earlier, and use the **yum** command during either Red Hat exam. Besides, it's an excellent tool for administering Red Hat systems.

Start with a simple command: **yum list**. It'll return a list of all packages, whether they're installed or available, along with their version numbers and repositories. If you want more information about a specific package, the **yum info** command can help. For example, the following command is functionally equivalent to **rpm -qi samba**:

```
# yum info samba
```

The **rpm -qi** command works if the queried package is already installed. The **yum info** command is not subject to that limitation.

Installation Mode

There are two basic installation commands. If you haven't installed a package before, or you want to update it to the latest stable version, run the **yum install *packagename*** command. For example, if you're checking for the latest version of the Samba RPM, the following command will update it or add it if it isn't already installed on the target system.

```
# yum install samba
```

Copyright © 2011 by The McGraw-Hill Companies

If you just want to keep the packages on a system up to date, run the **yum update *packagename*** command. For example, if you already have the Samba RPM installed, the following command makes sure it's updated to the latest version:

```
# yum update samba
```

If you haven't installed Samba, this command doesn't add it to your installed packages. In that way, the **yum update** command is analogous to the **rpm -F** command.

Of course, the **yum** command is not complete without options that can uninstall a package. The first one is straightforward, as it uninstalls the Samba package along with any dependencies.

```
# yum erase samba
```

The **yum update** command by itself is powerful; if you want to make sure that all installed packages are updated to the latest stable versions, run the following command:

```
# yum update
```

The **yum update** command may take some time as it communicates with the RHN or other repositories. It downloads the current database of packages with all dependencies. It then finds all packages with available updates, and adds them to the list of packages to be updates. It then finds all dependent packages if they're not already included in the list of updates.

What if you just want a list of available updates? The **yum list updates** command can help there. It's functionally equivalent to the **yum check-update** command.

But what if you aren't quite sure what to install? For example, if you want to install the Evince document reader, and think the operational command includes the term "evince," the **yum whatprovides */*evince*** command can help.

Alternatively, to search for all instances of files with the .repo extension, run the following command:

```
# yum whatprovides */*.repo
```

It lists all instances of the packages with files that end with the .repo extension, with the associated RPM package. The first wildcard is required, since the **whatprovides** option requires the full path to the file. It accepts partial filenames; for example, the **yum whatprovides /etc/init/*** command returns the RPM associated with files in the /etc/init directory. Once the needed package is known, you can proceed with the **yum install *packagename*** command.

Copyright © 2011 by The McGraw-Hill Companies

In many cases, problems with* yum *can be solved with the* yum clean all *command. If there are recent updates to RHN packages (or third-party repositories), this command flushes the current cache of headers, allowing you to resynchronize headers with configured repositories, without having to wait the default 90 minutes before the cache is automatically flushed (as defined by the* metadata_expire *directive in /etc/yum.conf).

Security and yum

Security can be a concern, especially with RPM packages downloaded over the Internet. If a cracker were to somehow penetrate the RHN, or perhaps a third-party repository, how would you know that packages from those sources were genuine? The answer is the GNU Privacy Guard (GPG) key, which is the open-source implementation of Pretty Good Privacy (PGP). It's the same system described earlier in this chapter for RPM packages. As an example, look at the output the first time a new package is installed over a network on RHEL 6:

```
# yum install samba
```

After packages are downloaded, you'll see something similar to the following messages:

```
warning: rpmts_HdrFromFdno: Header V3 RSA/SHA256 Signature, key ID f21541eb: NOKEY

 rhel/gpgkey      | 6.3 kB     00:00 ...
Importing GPG key 0xF21541EB "Red Hat, Inc. (release key 2) <security@redhat.com>"
from
/etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
Is this ok [y/N]: y
Importing GPG key 0x2FA658E0 "Red Hat, Inc. (auxiliary key) <security@redhat.com>"
from
/etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
Is this ok [y/N]: y
```

If you're simultaneously downloading packages from other repositories, additional GPG keys may be presented for approval. As suggested by the last line, **N** is the default response; you actually have to type in **y** to proceed with the download and installation; not only of the GPG key, but also of the package in question.

You may note that the GPG key used is from the same directory of keys associated with the **rpm** command earlier in this chapter. And that makes sense, as the **yum** command is essentially just a capable front end to the **rpm** command.

Copyright © 2011 by The McGraw-Hill Companies

Updates and Security Fixes

Red Hat maintains a public list of errata at http://rhn.redhat.com/errata/. Such errata are classified by RHEL releases. If you have an RHEL subscription, affected packages are normally made available through the RHN. All you need to do is run the **yum update** command periodically. This list is useful for those third parties who use RHEL source code, such as CentOS, Scientific Linux, or even Oracle Linux.

Package Groups and yum

The **yum** command can do more. It can install and remove packages in groups. These are the groups defined in the .xml files described in Chapter 2. One location for that file is on the RHEL 6 DVD, in the /repodata subdirectory. At the start of most of those stanzas, you'll see the **<id>** and **<name>** XML directives, which list two identifiers for each of those groups.

But that's a lot of work to find a package group. The **yum** command makes it simpler. With the following command, you can identify available package groups from configured repositories:

```
# yum grouplist
```

Note how the groups are divided into installed and available groups. Some of the groups listed may be of particular interest, such as "Remote Desktop Clients", some of which you'll use in Chapter 9. To find out more about this group, run the following command, with output shown in Figure 7-10.

```
# yum groupinfo "Remote Desktop Clients"
```

FIGURE 7-10 Packages in the Remote Desktop Clients group

```
[root@server1 yum.repos.d]# yum groupinfo "Remote Desktop Clients"
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process

Group: Remote Desktop Clients
 Optional Packages:
   rdesktop
   spice-client
   spice-xpi
   tigervnc
   tsclient
   vinagre
[root@server1 yum.repos.d]#
```

Copyright © 2011 by The McGraw-Hill Companies

Note how the packages are all listed as "Optional Packages". In other words, they're not normally installed with the package group. Thus, suppose you were to run the following command:

```
# yum groupinstall "Remote Desktop Clients"
```

Nothing would be installed. Desired packages from this package group have to be specifically named to be installed with commands like the following:

```
# yum install rdesktop
```

But optional packages are not the only category. The following command lists all packages in the Print Server package group, with output shown in Figure 7-11.

```
# yum groupinfo "Print Server"
```

Packages in this group are classified in two other categories. Mandatory packages are always installed with the package group. Default packages are normally installed with the package group; however, specific packages from this group can be excluded with the **-x** switch, unless changes are made during the RHEL installation process. For example, the following command installs the two mandatory and five default packages:

```
# yum groupinstall "Print Server"
```

FIGURE 7-11 Packages in the Print Server group

```
[root@server1 yum.repos.d]# yum groupinfo "Print Server"
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process

Group: Print Server
 Description: Allows the system to act as a print server.
 Mandatory Packages:
   cups
   printer-filters
 Default Packages:
   foomatic-db-ppds
   gutenprint
   gutenprint-cups
   hpijs
   paps
[root@server1 yum.repos.d]#
```

Copyright © 2011 by The McGraw-Hill Companies

In contrast, the following command excludes the paps and the gutenprint-cups packages from the list of those to be installed:

```
# yum groupinstall "Print Server" -x paps -x gutenprint-cups
```

The options to the **yum** command are not complete unless there's a command that can reverse the process. As suggested by the name, the **groupremove** option uninstalls all packages from the noted package group:

```
# yum groupremove "Print Server"
```

Exclusions are not possible with the **groupremove** switch. If you don't want to remove all packages listed in the output to the command, it may be best to remove target packages individually.

More yum Commands

A number of additional **yum**-related commands are available. Two of them may be of particular interest to those studying for the Red Hat exams: **yum-config-manager** and **yumdownloader**, which can display all current settings for each repository as well as download individual RPM packages. One more related command is **createrepo**, which can help you set up a local repository.

View All Directives with yum-config-manager

To some extent, the directives listed in the yum.conf and related configuration files provide only a small snapshot of available directives. To review the full list of directives, run the **yum-config-manager** command. Pipe it to the **less** command as a pager. It includes 100 lines. The excerpt from the [main] repository shown in Figure 7-12 is based on the connection to the RHN.

Many of the directives associated with **yum** are not filled in, such as **assumeyes**; some don't really matter, such as the color directives. Some of the other significant directives are shown in Table 7-5. It is not a comprehensive list. If you're interested in a directive not shown, it's defined in the man page for the yum.conf file.

Package Downloads with yumdownloader

As suggested by the name, the **yumdownloader** command can be used to download packages from yum-based repositories. It's a fairly simple command. For example, the

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-12

A partial list of yum directives

```
diskspacecheck = True
distroverpkg = redhat-release
enable_group_conditionals = True
enabled = True
enablegroups = True
errorlevel = 2
exactarch = True
exactarchlist =
exclude =
failovermethod = priority
gaftonmode =
gpgcheck = True
group_package_types = mandatory,
   default
groupremove_leaf_only =
history_record = True
history_record_packages = yum,
   rpm
http_caching = all
installonly_limit = 3
installonlypkgs = kernel,
   kernel-bigmem,
   kernel-enterprise,
   kernel-smp,
   kernel-debug,
   kernel-unsupported,
   kernel-source,
   kernel-devel,
   kernel-PAE,
```

following command reviews the contents of configured repositories for a package named cups.

```
# yumdownloader cups
```

Either the RPM package is downloaded to the local directory, or the command returns the following error messages:

```
No Match for argument cups
Nothing to download
```

Sometimes, more specifics are required. If there are multiple versions of a package stored on a repository, the default is to download the latest version of that package. That may not always be what you want. For example, if you want to use the originally released RHEL 6 kernel, the following command downloads the original RHEL 6 version of the kernel package:

```
# yumdownloader kernel-2.6.32-71.el6
```

Copyright © 2011 by The McGraw-Hill Companies

TABLE 7-5 Configuration Parameters from yum-config-manager

Configuration Directive in yum	Description
alwaysprompt	Prompts for confirmation on package installation or removal.
assumeyes	Set to no by default; if set to 1, **yum** proceeds automatically with package installation and removal.
cachedir	Set to the directory for database and downloaded package files.
distroverpkg	Refers to the /etc/redhat-release file with the name of the release.
enablegroups	Supports **yum group*** commands.
installonlypkgs	Lists packages that should never be updated; normally includes Linux kernel packages.
logfile	Specifies name of file with log information, normally /var/log/yum.log.
pluginconfpath	Notes the directory with plugins, normally /etc/yum/pluginconf.d.
reposdir	Specifies the directory with repository configuration files.
ssl*	Supports the use of the Secure Sockets Layer (SSL) for secure updates.
tolerant	Determines whether **yum** stops if an error is made in update package names.

Create Your Own Repository with createrepo

An earlier version of the RHCE objectives for RHEL 6 suggested that you should know how to "create a private yum repository." While that objective has since been removed, it's a logical future direction for the Red Hat exams.

Custom repositories can provide additional control. Enterprises who want to control the packages installed on their Linux systems can create their own customized repositories. While it can be based on the standard repositories developed for a distribution, it can include additional packages such as custom software unique to an organization. Just as easily, it can omit packages that may violate organizational policies such as games. Limits on the choices for certain functions such as browsers can minimize related support requirements.

To create a customized repository, you need to collect desired packages in a specific directory. The **createrepo** command can process all packages in that directory. The database is created in XML files in a repodata/ subdirectory. An example of this package database already exists in the repodata/ subdirectory of the RHEL 6 DVD.

The RHN enables support of customized repositories with related products, including the Red Hat Proxy Server and the Red Hat Satellite Server. For more

Copyright © 2011 by The McGraw-Hill Companies

information on repository management, see *Linux Patch Management*, written by this author, published by Prentice Hall.

CERTIFICATION OBJECTIVE 7.04

More Package Management Tools

Whether a system is connected to the RHN or remote repositories provided by a distribution like CentOS or Scientific Linux, it uses the same basic package management tools. Each of these alternatives uses the **rpm** command to process RPM packages. They use the **yum** command to satisfy dependencies and install groups of packages. And that makes sense, as the rebuild distributions are built on the same source code as RHEL 6.

Those similarities extend to GUI-based package management tools. While the identity of these tools have changed between RHEL 5 and RHEL 6, they're still front ends to the **rpm** and **yum** commands. They take advantage of the package groups configured in the .xml file described in Chapter 2. Since Red Hat uses GNOME as the default GUI desktop environment, the associated tools for RHEL 6 are based on that interface.

However, one thing that the rebuild distributions don't have is access to the RHN. That situation may only be temporary, Red Hat has recently started to release RHN software under open-source licenses. And developer groups are at work with projects like Spacewalk. That's important for the enterprise, as the RHN provides tools to administer groups of systems remotely from a single Web-based interface.

on the job

Subscriptions to the RHN includes access to the actual RHEL 6 operating system releases. Trial subscriptions support RHN access for that trial period. However, if you or your organization can't afford RHN subscriptions for every system, consider the tools provided by the Spacewalk project. It provides all of the functionality of the RHN, except for timely access to updates.

For RHEL 6, GUI-based package management tools are based on the PackageKit. However, it's quite possible that the PackageKit won't be available on a server, or perhaps even a system configured for a Red Hat exam. If you absolutely need the PackageKit, install the required RPMs with the **yum install gnome-packagekit**

Copyright © 2011 by The McGraw-Hill Companies

command. Of course, if you're already comfortable with the **yum** command, you may not need the PackageKit.

exam Watch

While the RHN is listed as part of the RHCSA objectives, it's listed in context as a choice. Whether you're installing or updating software packages from the RHN, "a remote repository, or a local filesystem," the rpm and yum commands are the same. Of course, it would be simplest if you did have an official subscription the RHN.

The GNOME Software Update Tool

If you're running a GUI in RHEL 6, the standard graphical software tool is based on the PackageKit application, configured for GNOME. It starts with the Software Update tool, which you can start from a GUI command line with the **gpk-update-viewer** command. Alternatively, from the GNOME Desktop Environment, click System | Administration | Software Update. The tool as shown in Figure 7-13 lists packages that are available for update.

It's a pretty straightforward interface. It's effectively a front end to the **yum update** command. Note the additional information, with a description of changes. Updates may be classified in up to six different categories, as shown in Figure 7-14, an excerpt from www.packagekit.org/pk-faq.html. The update categories shown in Figure 7-13 are enhancements, bug fixes, and security.

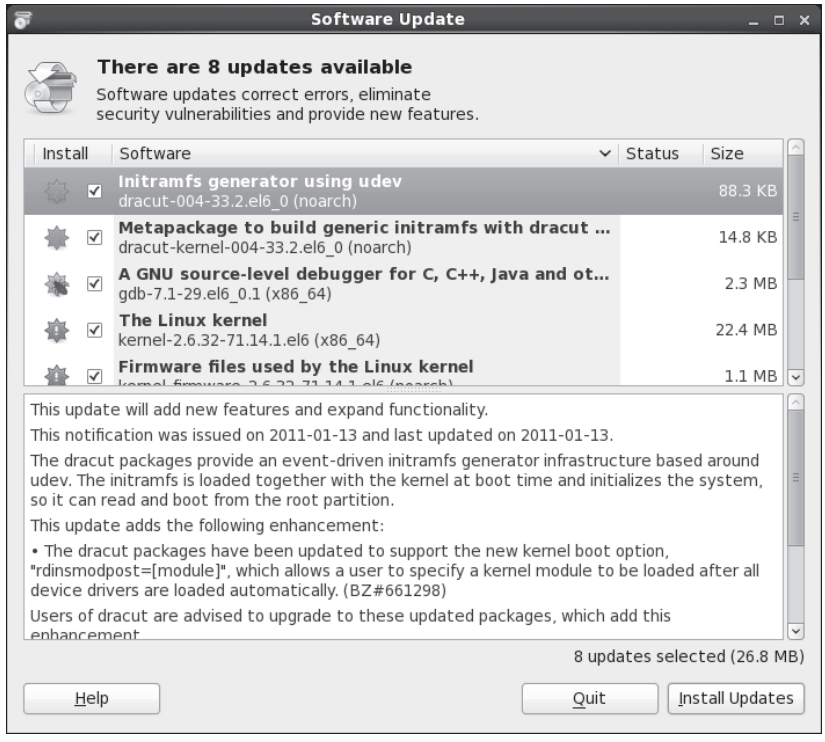
Automated Updates

It may be important to make sure the latest security updates are installed as quickly as possible. To that end, open the Software Update Preferences tool shown in Figure 7-15. You can open it by clicking System | Preferences | Software Updates, or from a GUI command line with the **gpk-prefs** command. You can configure the system to check for updates on an hourly, daily, or a weekly basis, or not at all. When updates are found, you can configure automatic installation of all available updates, of only security updates, or of nothing at all.

Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-13

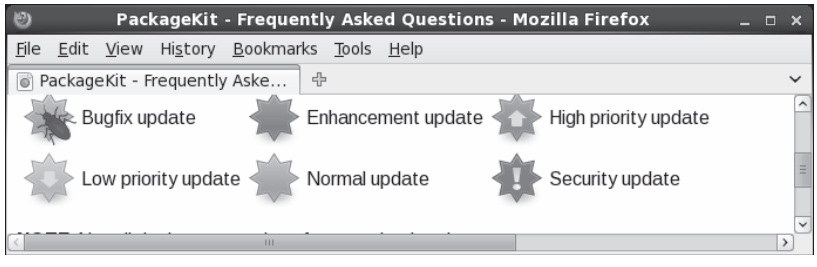
The GNOME Software Update tool



Changes made through the Software Preferences tool are stored in the authorized user's home directory, in a %gconf.xml file in the .gconf/apps/gnome-packagekit/update-icon subdirectory. Just be aware, changes may not be written to this file until the user logs out of the GUI.

FIGURE 7-14

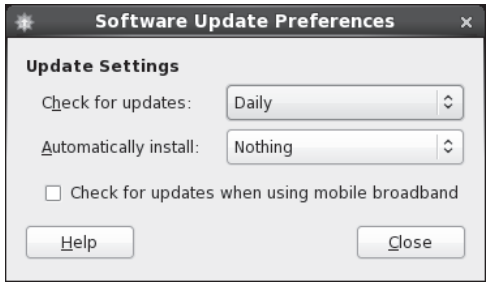
PackageKit update types



Copyright © 2011 by The McGraw-Hill Companies

FIGURE 7-15

The GNOME Software Update Preferences tool

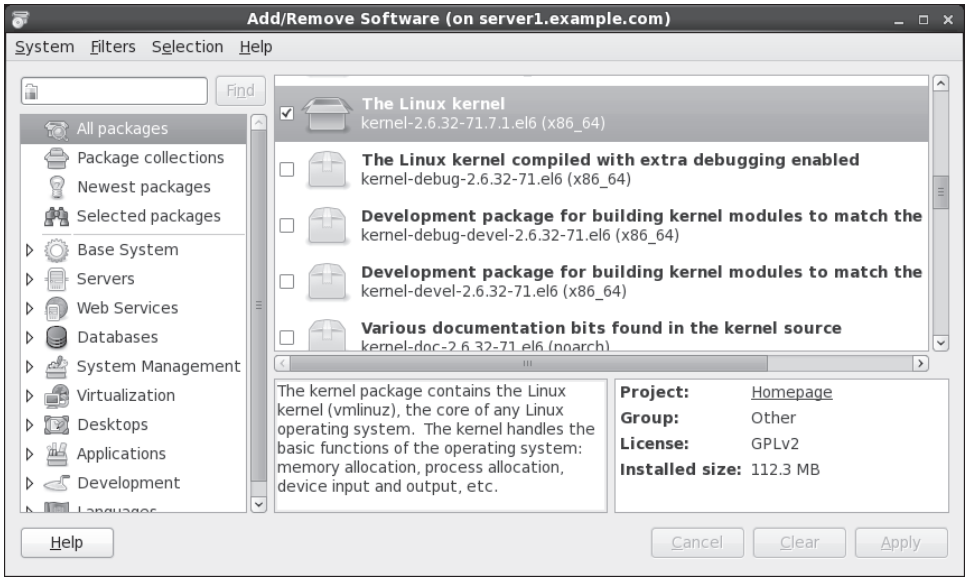


GNOME Add/Remove Software Tool

You can add, update, and remove packages with a graphical tool. To start the Add/Remove Software tool from a GUI command line, run the **gpk-application** command, or click System | Administration | Add/Remove Software. It opens the tool shown in Figure 7-16. Here you can install more than one package or package group at a time. Once packages are selected (or deselected), the tool automatically calculates dependencies and installs (or removes) them, along with the selected packages.

FIGURE 7-16

The GNOME Add/Remove Software tool



Copyright © 2011 by The McGraw-Hill Companies

You can use the Add/Remove Software tool to add the packages or package groups of your choice. In the upper-left part of the screen, there are four basic options:

- **All Packages** All packages from available repositories are listed in alphabetic order of RPM package name.
- **Package Collections** Options in this list are packages collected in groups; these are the same groups shown in the output to the **yum grouplist** command described earlier.
- **Newest Packages** Packages in this list do not include earlier versions.
- **Selected Packages** Packages in this list are in the queue for installation or removal, awaiting an in-process step such as downloads.

The package collections associated with the **yum grouplist** command are further subdivided in the lower-left part of the screen. When you select an individual group, every package member of the group is open for selection. That includes the mandatory, default, and optional packages described earlier in the XML package file described in Chapter 2; none of those packages are selected by default.

The options are straightforward. When a package or package group is selected or deselected for installation or removal, the Apply button becomes clickable. Once clicked, the tool uses the **yum** command to calculate dependencies. If there are no dependencies, the installation proceeds immediately. If there are dependencies, the entire list of packages to be installed or removed is presented for your approval.

EXERCISE 7-2

Installing More with yum and the Add/Remove Software Tool

This exercise requires a network connection to a remote repository, or at least a RHEL 6 DVD copied or mounted as a repository as configured earlier in this chapter. If you're using a rebuild of RHEL 6, you'll need to make sure the connection to the core repository is active, perhaps with a **ping** command to the URL of that repository as defined in the appropriate file in the /etc/yum.repos.d directory. Given the possible variations, exact steps are not possible there.

1. Run the **yum list** command. Assuming an active network connection and a responsive repository, you'll see a full list of available packages, including those already installed. Note the label in the right column; it will either show

Copyright © 2011 by The McGraw-Hill Companies

the repository where a package is available or note that the package is already installed.

2. Enter the **gpk-application** command in a GUI command line. This should open the Add/Remove Software tool.
3. In a second command line console, type in the **yum grouplist** command. In the Add/Remove Software tool, select Package Collections. Compare the list of package groups in each output.
4. Review available package groups in the Add/Remove Software tool. For example, click the arrow next to Servers. Under the options that appear, click CIFS Server. (CIFS stands for the Common Internet File System.) There's only one official package in the RHEL 6 configuration of this group. If you see more than one package, they are different versions of the same package. Select the latest available version of the package, which will be installed when you click Apply.
5. Click the All Packages option. Review the list of available packages. Packages that are already selected (or don't even have a check box) are currently installed. If you select or deselect a package, it will be installed or removed (with dependencies) when you click Apply.
6. Locate the text box in the upper-left corner of the Add/Remove Software tool. Type in a common search term such as *gnome* and watch as a long list of packages are shown. Compare the result to the output of the **yum search gnome** command.
7. Use a less common search term such as *iptables*. Highlight the iptables package and review it in the lower-right part of the screen. Compare the result to the output of the **yum info iptables** command.
8. Once you've selected some packages, click Apply. If there are dependencies, you'll see a window with a list of packages that you've selected for installation and removal. Depending on whether the packages are to be installed or removed, you'll see an Install or Remove button and a Cancel button.
9. Wait as downloads packages are installed. When finished, click System | Quit to exit from the Add/Remove Software tool.

Copyright © 2011 by The McGraw-Hill Companies

The Red Hat Network

Before you use the RHN to administer RHEL 6 systems, those systems must be registered. To that end, you'll need either a registration code or available entitlements on the subject RHN account. Alternatively, you can configure RHEL and rebuild (and even Fedora) systems on a Spacewalk server. For more information, see http://spacewalk.redhat.com/. Remember, the related objective suggests that all you need to know is how to install and update packages from the RHN. And that skill was already covered with the **rpm** and **yum** commands, along with the related GUI tools discussed in most of this chapter.

If you have a limited budget and can afford some RHEL subscriptions, it's technically feasible to set up mirrors of downloaded packages on a Spacewalk server. (I do not know whether such a mirror of binary Red Hat RPM packages would violate any agreements associated with a RHN subscription.) Of course, while Red Hat does sponsor Spacewalk, it does not include official support for that software. Alternatively, you could purchase supported access to RHN Proxy Server and Satellite Server products.

Perhaps the key benefit of the RHN or a substitute like Spacewalk is the ability to manage all RHEL and rebuild distribution systems remotely, over a Web-based interface. Once an appropriate connection is configured from the client systems, RHN can even run remote commands on any schedule. If you're administering a whole bunch of systems, RHN supports configuration of systems in groups. For example, if there are ten systems configured as RHEL 6-based web servers, you can configure those systems as a single group. You can then schedule a single command that's applied to all of those systems remotely.

The following list of capabilities highlight the features of RHN:

- Pre-scheduled commands
- Remotely installed packages
- The ability to edit and add custom configuration files
- Options to create Kickstart installations
- The ability to create snapshots

It also allows you to configure different subscription channels for each system, and more. For more information on the RHN, see the latest version of the reference guide, available from https://rhn.redhat.com/rhn/help/reference/.

Copyright © 2011 by The McGraw-Hill Companies

If a system isn't already registered, the following steps support registration of an RHEL 6 system from the command line:

1. Run **rhn_register** from the command line. If the system is already registered, you're prompted with the opportunity to leave the registration process.
2. You'll see a screen related to the configuration of software updates. The exact wording varies, depending on whether the registration is proceeding at the console or the GUI. If you need more information about the RHN, select Why Should I Connect To RHN; otherwise, select Next or Forward to continue. (Next is the option in the console; Forward is the option in the GUI.)
3. In the GUI version, you'll see options associated with the RHN Proxy and Satellite servers. The console version of the registration tool skips this step.
4. You'll see a screen where you can enter RHN login information. Do so and select Next or Forward to continue.
5. Now choose whether to register a system profile and whether to send basic hardware information about your system; make appropriate decisions and select Next or Forward to continue.
6. Next, choose to include a list of installed packages, which helps the RHN check whether you need software and security updates. Make any desired changes and select Next or Forward to continue.
7. Finally, choose whether to send your system profile to the RHN. If you click Cancel, the tool stops, and your system is not registered. If you want to register, select Next or Forward to continue.
8. Your system attempts to contact the RHN server (or possibly your RHN Satellite Server). If a free subscription entitlement is available, a message eventually appears that the system is successfully registered with the RHN.

CERTIFICATION SUMMARY

This chapter focuses on the management of RPM packages. With different switches, you also looked at how the **rpm** command installs, removes, and upgrade packages, as well as how it works locally and remotely. When presented with a new version of a kernel, it's important to never "upgrade." A properly configured installation of a later kernel version does not overwrite, but brings kernels together, side by side. You'll then be able to boot into either kernel.

Copyright © 2011 by The McGraw-Hill Companies

With the **rpm** command, you also learned how to query packages, to examine to which package a file belongs, to validate a package signature, to find the current list of installed RPMs. You also looked at the difficulties associated with dependencies, which drove developers to the **yum** command.

The **yum** command is, in part, a front end to the **rpm** command. When there are dependencies, it installs those packages simultaneously. You learned how to configure Red Hat and other repositories to work with the **yum** command. You should now be able to configure even the RHEL 6 DVD as its own repository. As you saw, the **yum** command also can install or remove package groups, as defined by the XML database file of packages on the RHEL 6 DVD. The **yum** command is fully compatible with the RHN.

While additional package management tools are available from the GUI, they are front ends to the **yum** and **rpm** commands. With the **gpk-update-viewer** command, you started the Software Update tool to identify and install available updates. With the **gpk-prefs** command, you started the Software Update Preferences tool that can check for and install security or all available updates on a regular schedule. With the **gpk-application** command, you opened the Add/Remove Software tool, which also can be used to add or remove packages and package groups. If you have an RHEL subscription, systems can also be kept up to date through the RHN's Web-based interface.

Copyright © 2011 by The McGraw-Hill Companies

TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 7.

The Red Hat Package Manager

- The RPM database tracks where each file in a package is located, its version number, and much more.
- The **rpm -i** command installs RPM packages.
- The **rpm -e** command uninstalls RPM packages.
- The **rpm** command can even install RPMs directly from remote servers.
- RPM package verification is supported by the GPG keys in the /etc/pki/rpm-gpg directory.
- Kernel RPMs should always be installed, never upgraded.
- The Upgrade mode of RPM replaces the old version of the package with the new one.

More RPM Commands

- The **rpm -q** determines whether packages are installed on a system; with additional switches, it can list more about a package and identify the package for a specific file.
- Package signatures can be checked with the **rpm --checksig** (or **-K**) command.
- The **rpm -V** command can identify files that have changed from the original installation of the package.
- The **rpm -qa** command lists all currently installed packages.

Dependencies and the yum Command

- By including additional required packages, the **yum** command can help avoid "dependency hell."
- The behavior of the **yum** command is configured in the /etc/yum.conf file, plugins in the /etc/yum/pluginconf.d directory, and repositories configured in the /etc/yum.repos.d directory.

Copyright © 2011 by The McGraw-Hill Companies

- ❑ Red Hat organizes packages in several different repositories for RHEL 6.
- ❑ Repositories for rebuild distributions and from third parties are accessible online.
- ❑ The **yum** command can install, erase, and update packages. It also can be used to search in different ways.
- ❑ The **yum** command uses the GPG keys developed for RPM packages.
- ❑ The **yum** command can install, remove, and list package groups.

More Package Management Tools

- ❑ RHEL 6 package management tools are based on the PackageKit, built for GNOME.
- ❑ With PackageKit tools, you can install and remove packages and package groups.
- ❑ The PackageKit also includes tools focused on current updates. It can also set up updates of security or all packages on a schedule.
- ❑ The RHN or Spacewalk can help you manage subscribed systems remotely using a Web-based interface.

Copyright © 2011 by The McGraw-Hill Companies

SELF TEST

The following questions will help measure your understanding of the material presented in this chapter. As no multiple-choice questions appear on the Red Hat exams, no multiple-choice questions appear in this book. These questions exclusively test your understanding of the chapter. It is okay if you have another way of performing a task. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of these questions.

The Red Hat Package Manager

1. What command would you use to install the penguin-3.26.i386.rpm package, with extra messages in case of errors? The package is on the local directory.

   _____

2. What command would you use to upgrade the penguin RPM with the penguin-3.27.i386.rpm package? The package is on the ftp.remotemj02.abc server.

   _____

3. If you've downloaded a later version of the Linux kernel to the local directory, and the package filename is kernel-2.6.32-100.el6.x86_64.rpm, what's the best command to make it a part of your system?

   _____

4. What directory contains GPG keys on an installed system?

   _____

More RPM Commands

5. What command lists all currently installed RPMs?

   _____

6. What file lists the RPMs installed during the system installation process?

   _____

7. If you've downloaded an RPM from a third party called third.i386.rpm, how can you validate the associated package signature?

   _____

Copyright © 2011 by The McGraw-Hill Companies

Dependencies and the yum Command

8. What is the full path to the directory where the location of **yum** repositories are normally configured?

   _____

9. What command searches **yum** repositories for the package associated with the /etc/passwd file?

   _____

More Package Management Tools

10. What command-line command lists the package groups shown in the Add/Remove Software tool?

    _____

11. Name two allowable time periods for automatic updates, as defined by the Software Update Preferences tool.

    _____

    _____

12. What command from the console starts the process of registration on the RHN?

    _____

LAB QUESTIONS

Red Hat presents its exams electronically. For that reason, most of the labs in this chapter are available from the CD that accompanies the book, in the Chapter7/ subdirectory. They're available in .doc, .html, and .txt format, to reflect standard options associated with electronic delivery on a live RHEL 6 system. In case you haven't yet set up RHEL 6 on a system, refer to the first lab of Chapter 2 for installation instructions. The answers for each lab follows the Self Test answers for the fill-in-the-blank questions.

Copyright © 2011 by The McGraw-Hill Companies

SELF TEST ANSWERS

The Red Hat Package Manager

1. The command that installs the penguin-3.26.i386.rpm package, with extra messages in case of errors, from the local directory, is

```
# rpm -iv penguin-3.26.i386.rpm
```

   Additional switches that don't change the functionality of the command, such as **-h** for hash marks, are acceptable. This applies to subsequent questions as well.

2. The command that upgrades the aforementioned penguin RPM with the penguin-3.27.i386.rpm package from the ftp.remotemj02.abc server is

```
# rpm -Uv ftp://ftp.remotemj02.abc/penguin-3.26.i386.rpm
```

   If you use the default vsFTP server, the package may be in the pub/Packages/ subdirectory. In other words, the command would be

```
# rpm -Uv ftp://ftp.remotemj02.abc/pub/Packages/penguin-3.26.i386.rpm
```

   Yes, the question is not precise. But that's what you see in real life.

3. If you've downloaded a later version of the Linux kernel to the local directory, and the package filename is kernel-2.6.32-100.el6.x86_64.rpm, the best way to make it a part of your system is to install it—and not upgrade the current kernel. Kernel upgrades overwrite existing kernels. Kernel installations allow kernels to exist side by side; if the new kernel doesn't work, you can still boot into the working kernel. Since the desired package is already downloaded, you'd use a command similar to the following:

```
# rpm -iv kernel-2.6.32-100.el6.x86_64.rpm
```

   Variations of the **rpm** command, such as **rpm -i** and **rpm -ivh**, are acceptable. However, variations that upgrade, with the **-U** or **-F** switches, are incorrect.

4. The directory with GPG keys on an installed system is /etc/pki/rpm-gpg. The GPG keys on the RHEL 6 CD/DVD are not "installed" on a system.

More RPM Commands

5. The command that lists all installed RPMs is

```
# rpm -qa
```

Copyright © 2011 by The McGraw-Hill Companies

6. The file that lists the RPMs installed when you first installed the local system is /root/install.log. The /var/log/rpmpkgs file is complete but is updated only once per day and is available only if the rpm-cron package is installed.
7. If you've downloaded an RPM from a third party, call it third.i386.rpm, you'll first need to download and install the RPM-GPG-KEY file associated with that repository. You can then validate the associated package signature with a command like (note the uppercase **-K**); **--checksig** is equivalent to **-K**.

```
# rpm -K third.i386.rpm
```

Dependencies and the yum Command

8. The **yum** command repositories are normally configured in files in the /etc/yum.repos.d directory. Technically, **yum** command repositories can also be configured directly in the /etc/yum.conf file.
9. The **yum whatprovides /etc/passwd** command identifies packages associated with that file.

More Package Management Tools

10. This is a slightly tricky question, as the **yum grouplist** command lists the package groups also shown in the Add/Remove Software tool.
11. Allowable time periods for updates, as defined by the Software Update Preferences tool, are hourly, daily, and weekly.
12. The **rhn_register** command starts the process of registering a system on the RHN.

LAB ANSWERS

Lab 1

When complete, run the following commands to verify the connection:

```
# yum clean all
# yum update
```

The output should include output similar to the following:

```
Loaded plugins: refresh-packagekit
inst                | 3.7 kB     00:00 ...
inst/primary_db     | 2.3 MB     00:00 ...
Setting up Update Process
```

Copyright © 2011 by The McGraw-Hill Companies

This output verifies a successful connection to the FTP server. If you see something significantly different, check the following in the /etc/yum.repos.d/file.repo file:

- Make sure the stanza in this file starts with **[inst]**.
- Check the URL associated with the **baseurl** directive. It should match the URL of the FTP server defined in Chapter 1, Lab 2. You should be able to run the **lftp** or **ftp** commands with that URL from a command line interface. If that doesn't work, either the FTP server is not running, or messages to that server are blocked by a firewall.
- If there were problems, fix them. Then try the previous commands again.

Lab 2

One way to check all of the files in the /usr/sbin directory is to use the **rpm -Va | grep /usr/sbin** command.

If successful, you'll identify the /usr/sbin/vsftpd and /etc/vsftpd/vsftpd.conf files as different from their original versions as installed from the RPM. Changes to a configuration file are not a big deal, especially if it's been customized in any way. However, changes to the binary file are a reason for suspicion.

Assuming standard Red Hat RPM packages, removal and reinstallation should preserve changes to the vsftpd.conf file in a vsftpd.conf.rpmsave file.

If you really do have a security concern, additional measures are appropriate. For example, some security professionals might compare all files on a suspect system to the files on a verified baseline system.

In that case, it may be simplest to take a copy or clone of the baseline system, reinstall the vsftpd RPM, and reconfigure it as needed. Assuming the baseline system is secure, you'd then be reasonably sure the new server would also be secure.

The changes made by the script to this lab set a new modification time for the /usr/sbin/vsftpd binary and appended a comment to the end of the vsftpd.conf configuration file. If you want to restart with fresh copies of these packages, back up your current vsftpd.conf file and run the **rpm -e vsftpd** command to uninstall the package. If the RPM package has been properly configured, you should see at least the following warning message:

```
warning: /etc/vsftpd/vsftpd.conf saved as /etc/vsftpd/vsftpd.conf.rpmsave
```

You can then reinstall the original package from either the installation DVD or a remote repository. Alternatively, you could delete (or move) the changed files and then run the following command to force the **rpm** command to provide the original copies of these files from the associated package. The version number is based on the RHEL 6.0 DVD.

```
# rpm -ivh --force vsftpd-2.2.2-6.el6.x86_64.rpm
```

Copyright © 2011 by The McGraw-Hill Companies

Lab 3

This lab is intended to help you examine what the **yum update** command can do. It's the essential front end to GUI update tools. As you can see from the update.txt file created in this lab, the messages display how **yum** appears for all newer packages from configured repositories or the RHN, downloads their headers, and uses them to check for dependencies that also need to be downloaded and installed.

Lab 4

This lab should be straightforward, as it involves the use of the Software Update Preferences tool, which you can start from a GUI command line with the **gpk-prefs** command. If successful, you'll see these changes in the %gconf.xml file, in the .gconf/apps/gnome-packagekit/update-icon subdirectory of your home directory. For example, the following excerpt suggests that updates are made every 86400 seconds, which corresponds to 24 hours.

```
<entry name="frequency_get_updates" mtime="1299287329" type="int" value="86400">
```

Bonus bit: if you are familiar with Linux, you may recognize the **mtime** as the number of seconds since the Unix epoch of January 1, 1970. To find the actual date associated with that modification time, run the following command:

```
# date -ud @1299287328
```

Lab 5

This lab is somewhat self-explanatory and is intended to help you explore what happens when you properly install a new kernel RPM. As with other Linux distributions, when you install (and do not use upgrade mode) for a new kernel, two areas are affected.

The new kernel is added as a new option in the GRUB configuration menu. The existing kernel should be retained as an option in that menu. When you reboot the system, try the new kernel. Don't hesitate to reboot the system again, and try the other option, probably the older kernel.

When you review the /boot directory, all of the previously installed boot files should be there. The new kernel RPM should add matching versions of all of the same files—with different revision numbers.

To keep this all straight, it helps if you made copies of the original versions of the GRUB configuration file and the file list in the /boot directory. If you choose to retain the newly installed kernel, great. Otherwise, uninstall the newly installed kernel. This is one case where revision numbers are required with the **rpm -e** command; the following is based on the removal of the kernel and kernel-firmware packages, based on version number 2.6.32-71.14.1el6:

```
# rpm -e kernel- 2.6.32-71.14.1el6.x86_64
# rpm -e kernel-firmware-2.6.32-71.14.1el6
```

Copyright © 2011 by The McGraw-Hill Companies

If the revision number of the kernel or kernel-firmware package that you installed during this lab is different, adjust the commands accordingly.

Lab 6

This lab is designed to give you practice with both the **yum** command and the Add/Remove Software tool. It should help you prepare for Chapter 9, and provide the skills required to install services for other chapters. You should realize by now that since all packages in the Remote Desktop Clients package group are optional, the **yum groupinstall "Remote Desktop Clients"** command doesn't install anything. You'll need to install each of the optional packages by name.

To identify the names of the packages to be installed, run the **yum groupinfo "Remote Desktop Clients"** command. Be sure to install every package from that group on both systems. The best method is with the **yum install *package1 package2*** ... command, where package1, package2, and so on, are names of packages in the "Remote Desktop Clients" package group.

Copyright © 2011 by The McGraw-Hill Companies