CHAPTER 8

Managing Software and System Resources

IN THIS CHAPTER

This chapter introduces concepts, procedures, and software you can use to manage installed system resources on a Fedora Core Linux system. Managing the system resources—including software, storage, and memory—of your computer is important for a number of reasons. Good resource management promotes and supports efficient, productive sessions, a stable system, and satisfied users. Managing these resources involves installing, removing, upgrading, or rebuilding software packages—each a vital task that can contribute to your system's security.

As a Fedora Core Linux system administrator, you should also know how to maximize your system's resources by managing memory and storage for the most efficient system use. Properly managing your system resources ensures that you won't run out of room for new software, and you can expand the system to fit changing needs and new projects. In other words, system resource management is essential to providing the best possible computing experience for your users.

Fedora Core Linux provides a variety of tools for system resource management. The following sections introduce the RPM Package Manager (RPM), along with command-line and graphical software-management tools. You'll also learn about monitoring and managing memory and disk storage on your system.

Using RPM for Software Management

RPM was derived (in part) from early Linux package management software—named RPP, PMS, and PM—that were written in Perl. RPM was first used with Red Hat Linux 2.0 in late 1995, and then rewritten in C for the Red Hat Linux 3.0.3

(Picasso) release in 1996. Since then, the rpm command has been the prime feature of Red Hat's unique software management system, which is based on the concept of pristine sources, or the capability to use a single, initial archive of a program's source code to build packages for different systems and to track versions. With the release of Red Hat 8.0 (Psyche) in 2002, Red Hat offered a slightly updated graphical management interface for its venerable RPM application.

In addition to improving the package management of early software management scripts, RPM version 4.1 introduced software features designed to ease the task of building software for different platforms from a single set of source-code files. Changes can be tracked and kept outside a developer's initial source code and multiple packages can be built from scratch and installed at the same time—simultaneously, RPM also verifies installation dependencies. Additional features, such as a checksum and GNU Privacy Guard (GPG) signatures, enable binary software packages to be safely distributed without the fear of virus infection or the inclusion of Trojan code.

The rpm command uses the RPM system to install, remove (erase), upgrade, verify, and build software archives known as .rpm files. These archives, or packages, contain package identification (a signature), checksums (mathematically derived validation values), and an archive of the software, either in source or binary form. A .rpm package also contains quite a bit of additional information, such as a name, version, and basic description, and can include pre- and post-installation scripts used for software installation, erasure, or upgrading.

The RPM database installed on your computer keeps track of which versions of which packages are installed. RPM uses your system's /var/lib/rpm directory to store files (actually databases) containing information about the software installed on your system. You can use the ls command to view these files (you might see file sizes different from those shown here, depending on the amount of software you have installed):

```
$ ls -l /var/lib/rpm
total 53820
-rw-r--r--    1 rpm      rpm       5423104 Oct 14 19:53 Basenames
-rw-r--r--    1 rpm      rpm         12288 Oct 14 12:32 Conflictname
-rw-r--r--    1 root     root        16384 Oct 14 17:31 __db.001
-rw-r--r--    1 root     root      1318912 Oct 14 17:31 __db.002
-rw-r--r--    1 root     root       458752 Oct 14 17:31 __db.003
-rw-r--r--    1 rpm      rpm       1179648 Oct 14 19:53 Dirnames
-rw-r--r--    1 rpm      rpm       5521408 Oct 14 19:53 Filemd5s
-rw-r--r--    1 rpm      rpm         24576 Oct 14 19:53 Group
-rw-r--r--    1 rpm      rpm         20480 Oct 14 19:53 Installtid
-rw-r--r--    1 rpm      rpm         45056 Oct 14 19:53 Name
-rw-r--r--    1 rpm      rpm      41070592 Oct 14 19:53 Packages
-rw-r--r--    1 rpm      rpm        348160 Oct 14 19:53 Providename
-rw-r--r--    1 rpm      rpm         98304 Oct 14 19:53 Provideversion
```

```
-rw-r--r--    1 rpm      rpm         12288 Oct 14 19:53 Pubkeys
-rw-r--r--    1 rpm      rpm        237568 Oct 14 19:53 Requirename
-rw-r--r--    1 rpm      rpm        176128 Oct 14 19:53 Requireversion
-rw-r--r--    1 rpm      rpm         94208 Oct 14 19:53 Sha1header
-rw-r--r--    1 rpm      rpm         49152 Oct 14 19:53 Sigmd5
-rw-r--r--    1 rpm      rpm         12288 Oct 14 19:53 Triggername
```

The primary database of installed software is contained in the file named Packages. As you can see from the preceding example, this database can grow to 33MB (and perhaps larger) if you perform a full installation of Fedora Core Linux (more than 4GB of software). After you install Fedora Core Linux, `rpm` and related commands will use this directory during software management operations.

Command-Line and Graphical RPM Clients

As a Fedora Core Linux system administrator, you'll use the `rpm` command or the Fedora Core graphical clients to perform one of five basic tasks. These operations, which must be conducted by the root operator, include the following:

- Installing new software
- Erasing or removing outdated or unneeded packages
- Upgrading an installed software package
- Querying to get information about a software package
- Verifying the installation or integrity of a package installation

The `rpm` command has more than 60 different command-line options, but its administrative functions can be grouped according to the previous five types of action. Graphical RPM clients provide easy-to-use interfaces to these operations. As a system administrator, you'll have a choice between using a graphical interface and using `rpm`'s various command-line options. The general format of an `rpm` command is

```
# rpm option packagename
```

The basic options look like this:

- `-i`—Installs the selected package or packages.
- `-e`—Erases (removes) the selected package or packages.
- `-U`—Removes the currently installed package, and then installs software with the contents of the selected package or packages, leaving the existing configuration files.
- `-q`—Queries the system or selected package or packages.
- `-V`—Verifies installed or selected package or packages.

TWO HANDY OPTIONS

By appending `vh` to any option, you get

`v`Some status feedback.

`h`Hash marks as the work proceeds.

Many additional options can also be added to or used in conjunction with these options. These are summarized in the following table.

Option	Used To
`rpm-i`	Install a Package
Useful options to `-i`:	
`--excludedocs`	Doesn't install documentation to save space.
`--replacepkgs`	Replaces the package with a new copy of itself.
`--force`	The "big hammer"—Ignores all warnings and installs anyway.
`--noscripts`	Doesn't execute any pre- or post-install scripts.
`--nodeps`	Ignores any dependencies.
`--root` *`path`*	Sets an alternative root to *`path`*.
`rpm -e`	Erases (deletes) a Package.
Useful options to `-e`:	
`--nodeps`	Ignores any dependencies.
`rpm -U`	Upgrades a package, removing the older one but keeping modified files, such as configurations.
Useful options to `-U`:	
`--oldpackage`	Permits downgrading to an older version.
Other options are the same as with `rpm -i`.	
`rpm -q`	Queries about Package Information
Useful options to `-q`:	
`-p` *`file`*	Displays all information about the package *`file`*.
`-f` *`file`*	What package owns the file *`file`*?
`--whatprovides` *`x`*	Determines what packages provide *`x`*.
`--whatrequires` *`x`*	Determines what packages require *`x`*.
`-i`	Summarizes the package information.
`-l`	Lists the files in package.
`--scripts`	Displays the contents of any install, uninstall, or verifies scripts.
`--provides`	Displays the capabilities package provides.

`--requires`	Displays the capabilities package requires.
`rpm -V`	Verifies Packages Against the RPM Database
Useful options to `-V`:	
`-a`	Verifies all installed packages.
`rpm -K`	Uses GPG to verify a downloaded package.
Useful options to `-K`:	
`--nosignature`	If you lack public GPG encryption keys, do not have GPG installed, or are legally prohibited from using GPG, this still verifies the package using size and MD5 checksums.

Details on obtaining the Fedora Core public GPG encryption key and using it are at

`http://www.rpm.org/max-rpm/s1-rpm-checksig-using-rpm-k.html.`

RPM IS FOR PROGRAMMERS, TOO!

Remember that RPM was created not only to provide an easy to use administrative tool, but also as a developer's tool for use in multi-platform source-code package management. Programmers using `rpm` for development and distribution will use its `rpmbuild` command, along with a myriad of additional command-line flags. RPM can be used to build binaries, execute programs, test installations, verify and sign packages, build source packages, track versions, and target builds for specific architectures. Details can be found at the RPM home page (listed in the "Reference" section at the end of this chapter).

Using `rpm` on the Command Line

Because the new graphical RPM client can only install and uninstall RPM packages (for now—more functionality is promised), you will still end up administering RPM packages from the command line. You can perform all the five basic `rpm` operations using the `rpm` command from the command line. This section gives you an introduction to performing those operations. It also provides examples of how to install, verify, query, remove, and upgrade a software package.

The most common `rpm` operation is software installation. Using `rpm` is an easy way to keep track of installed software, and it can be used to quickly remove undesired packages. Use the `-i` option, along with the full or partial name (using regular expressions) of a software package, to install software with `rpm`. For example, to install the `unace` archiving package, use the `rpm` command like this:

```
# rpm -ivh http://mirrors.zoreil.com/plf.zarb.org/9.1/
➥i586/unace-2.2-2plf.i586.rpm
Retrieving http://mirrors.zoreil.com/plf.zarb.org/9.1/
➥i586/unace-2.2-2plf.i586.rpm
```

```
warning: /var/tmp/rpm-xfer.48amVs: V3 DSA signature: NOKEY, key ID 8df56d05
Preparing...                ########################################### [100%]
   1:unace
########################################### [100%]
```

This example uses the v and h options, which provide a more verbose output and display of hash marks to show the progress of the installation. The example also demonstrates the capability of rpm to use HTTP or FTP servers to fetch files for installation. It also shows that rpm can use GPG keys to validate a file. (The key was not installed in our example.)

You can also use rpm to query its database after installing packages to verify an installation. Use the -V option, along with the name of a software package, to verify installation your system. For example, to verify the unace archiving package, use the rpm command like this:

```
# rpm -V unace
```

NOTE

If everything is correct with your software installation, your system will display no response to rpm -V after you run the command; only problems are displayed.

As you can see from the following program output, you can get additional information about a package by adding additional verification options (such as two more v's) to the -V option. To get more information about an installed package, use one or more forms of the rpm query options. For example, to display concise information about an installed package, use the -q option, along with the i option and the installed package name, like this (note that your version will be different from that shown here):

```
# rpm -qi unace

Name        : unace                        Relocations: (not relocateable)
Version     : 2.2                               Vendor: Penguin Liberation Front
Release     : 2plf                          Build Date: Sat 01
➥Mar 2003 12:13:48 PM EST
Install date: Tue 02 Sep 2003 03:46:28 PM EDT      Build Host:
➥baader.subversion.alt
Group       : Archiving/Compression         Source RPM: unace-2.2-2plf.src.rpm
Size        : 401368                           License: freeware
Packager    : Guillaume Rousse <guillomovitch@zarb.org>
URL         : http://www.winace.com
Summary     : Decompressor for .ace format archives
Description :
Unace is a utility to extract, view, and test the contents of an ACE archive.
```

This form of the rpm query provides quite a bit of information about the software package. (You can also query packages before installation by providing a pathname for them.)

If this package isn't up-to-date, you can easily and quickly upgrade the package by downloading a newer version and then using rpm's -U or upgrade option like this:

```
# rpm -Uvh unace-2.2-2plf.i586.rpm
Preparing...                ########################################### [100%]
   1:unace                   ########################################### [100%]
```

Note that it wasn't necessary to remove the currently installed software package—the U option removes the old version of the software (saving the old configuration files), and then automatically installs the new software.

You can also upgrade your system software by using the rpm command's -F or "freshen" option, which will fetch a designated package from a remote FTP or HTTP server. For example, to upgrade the fetchmail-conf package, use rpm like this:

```
# rpm -Fv ftp://ftp.tux.org/linux/redhat/updates/9/en/os/i386/\
initscripts-7.14-1.i386.rpm
Retrieving ftp://ftp.tux.org/linux/redhat/updates/9/en/os/i386/\
initscripts-7.14-1.i386.rpm
Preparing packages for installation...
initscripts-7.14-1
```

Use the -e option, along with the name of a software package, to remove or erase software from your system with rpm. For example, to remove the unace archiving package, use the rpm command like this:

```
# rpm -e unace
```

Note that if the operation succeeds, no messages will be displayed on your system. You can quickly search for the names of installed packages by piping the output of rpm -qa through the grep and sort commands (see Chapter 5, "First Steps with Fedora," for additional information on grep and sort); here's how to do that search:

```
# rpm -qa ¦ grep mail ¦ sort
fetchmail-6.2.0-7

mailcap-2.1.14-1.1
mailx-8.1.1-31.1
mozilla-mail-1.4.1-8
procmail-3.22-11
sendmail-8.12.10-1.1
sendmail-cf-8.12.10-1.1
```

This example returns a sorted list of all packages with names containing the word mail.

NOTE

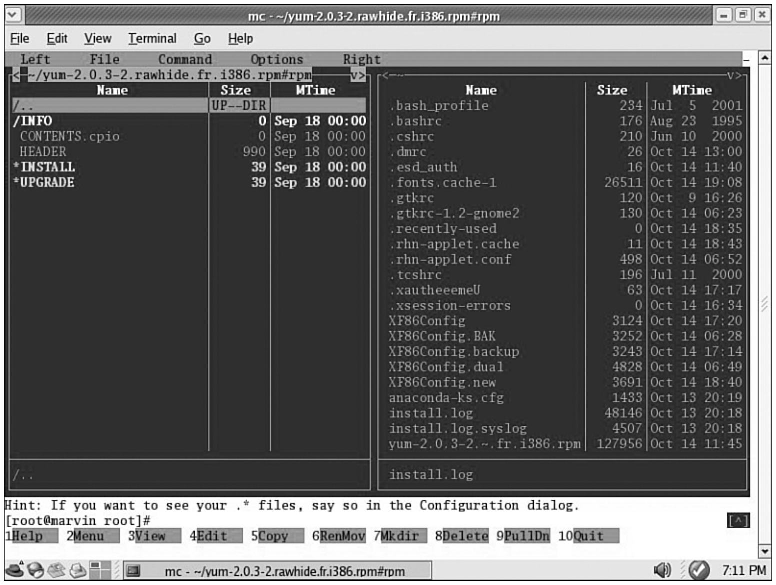
Another essential feature of the `rpm` command is its `--rebuilddb` option. If your system's RPM database becomes corrupted, this is your first (and perhaps only) option for restoring software management services. We hope that you never have to use this option; help ensure that by always backing up your data!

Package Organization with RPM

Software packages on your Fedora Core Linux system are organized into various groups, as you see later in this chapter. Using a group organization helps Fedora Core keep software organized by category and provides for hierarchical listings of software when using graphical RPM clients. You never have to manipulate these groups, but understanding the concept of package organization can help you gain familiarity with the way Fedora Core Linux works.

Extracting a Single File from an RPM File

Occasionally, it is useful to extract a single file from an RPM package. You can do so using the command-line version of `mc`, the Midnight Commander. In Figure 8.1, the Midnight Commander is displaying the contents of the `yum .rpm` file. The Midnight Commander is a UNIX clone of the famous DOS Norton Commander, a file management utility. Using `mc`, just highlight the RPM file and press Enter; the contents of the RPM file will be displayed. In the listing, you can browse the file structure of the RPM file and use `mc` to copy files from it.


mc - ~/yum-2.0.3-2.rawhide.fr.i386.rpm#rpm
File Edit View Terminal Go Help
mc - ~/yum-2.0.3-2.rawhide.fr.i386.rpm#rpm
7:11 PM

```
 Left     File     Command     Options     Right
<-~/yum-2.0.3-2.rawhide.fr.i386.rpm#rpm----v>  <-~----------------------------------------v>
        Name          | Size  |   MTime         Name                    |  Size  |   MTime
/..                   |UP--DIR|              .bash_profile              |    234|Jul  5  2001
/INFO                 |     0 |Sep 18 00:00  .bashrc                    |    176|Aug 23  1995
 CONTENTS.cpio        |     0 |Sep 18 00:00  .cshrc                     |    210|Jun 10  2000
 HEADER               |   990 |Sep 18 00:00  .dmrc                      |     26|Oct 14 13:00
*INSTALL              |    39 |Sep 18 00:00  .esd_auth                  |     16|Oct 14 11:40
*UPGRADE              |    39 |Sep 18 00:00  .fonts.cache-1             |  26511|Oct 14 19:08
                      |       |              .gtkrc                     |    120|Oct  9 16:26
                      |       |              .gtkrc-1.2-gnome2          |    130|Oct 14 06:23
                      |       |              .recently-used             |      0|Oct 14 18:35
                      |       |              .rhn-applet.cache          |     11|Oct 14 18:43
                      |       |              .rhn-applet.conf           |    498|Oct 14 06:52
                      |       |              .tcshrc                    |    196|Jul 11  2000
                      |       |              .xautheeemeU               |     63|Oct 14 17:17
                      |       |              .xsession-errors           |      0|Oct 14 16:34
                      |       |              XF86Config                 |   3124|Oct 14 17:20
                      |       |              XF86Config.BAK             |   3252|Oct 14 06:28
                      |       |              XF86Config.backup          |   3243|Oct 14 17:14
                      |       |              XF86Config.dual            |   4828|Oct 14 06:49
                      |       |              XF86Config.new             |   3691|Oct 14 18:40
                      |       |              anaconda-ks.cfg            |   1433|Oct 13 20:19
                      |       |              install.log                |  48146|Oct 13 20:18
                      |       |              install.log.syslog         |   4507|Oct 13 20:18
                      |       |              yum-2.0.3-2.~.fr.i386.rpm  | 127956|Oct 14 11:45

/..                                          install.log
Hint: If you want to see your .* files, say so in the Configuration dialog.
[root@marvin root]#                                                              [^]
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

FIGURE 8.1 A classic two-panel directory view and drop-down menus betray Midnight Commander's DOS inspiration, but it's a full-featured Linux file manager.

You might want to know what a `.rpm` script will do before you install the application. You can use the `F3` key in `mc` to view the script files. If you want to look at the scripts without using `mc`, use this command:

```
# rpm -q --scripts filename > scripts.txt
```

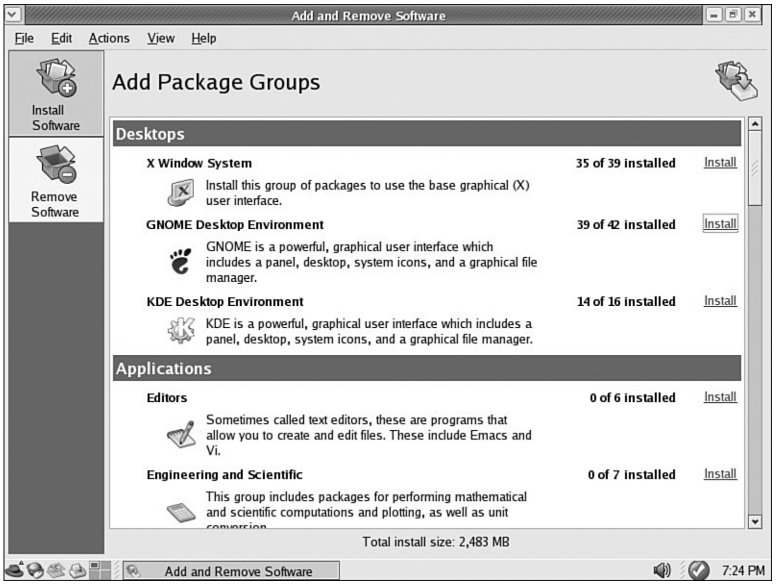
This command pipes the scripts into a file, where you can examine it with a text editor. You could also pipe it to the `less` pagination command to view the scripts on your display:

```
# rpm -q --scripts filename ¦ less
```

Graphical Package Management

The Fedora Core Linux graphical package management tool identified in the menu as Add/Remove Software is actually an application named `system-config-packages`. This application replaces `kpackage`, `gnorpm`, and `xrpm`—all of which are no longer provided. Add/Remove Software allows you to select packages arranged in categories and install or remove them. With the addition of YUM to Fedora Core, you may now add your own packages to the Fedora Core graphical tools' database, improving its usefulness over earlier versions.

Launch the Fedora Core GUI package manager by clicking the Start button on your desktop, and then choose System Settings, Add/Remove Applications. The package management tool launches with the Add and Remove Software screen, shown in Figure 8.2.



Add Package Groups

Desktops

X Window System — 35 of 39 installed — Install
Install this group of packages to use the base graphical (X) user interface.

GNOME Desktop Environment — 39 of 42 installed — Install
GNOME is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

KDE Desktop Environment — 14 of 16 installed — Install
KDE is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

Applications

Editors — 0 of 6 installed — Install
Sometimes called text editors, these are programs that allow you to create and edit files. These include Emacs and Vi.

Engineering and Scientific — 0 of 7 installed — Install
This group includes packages for performing mathematical and scientific computations and plotting, as well as unit conversion

Total install size: 2,483 MB

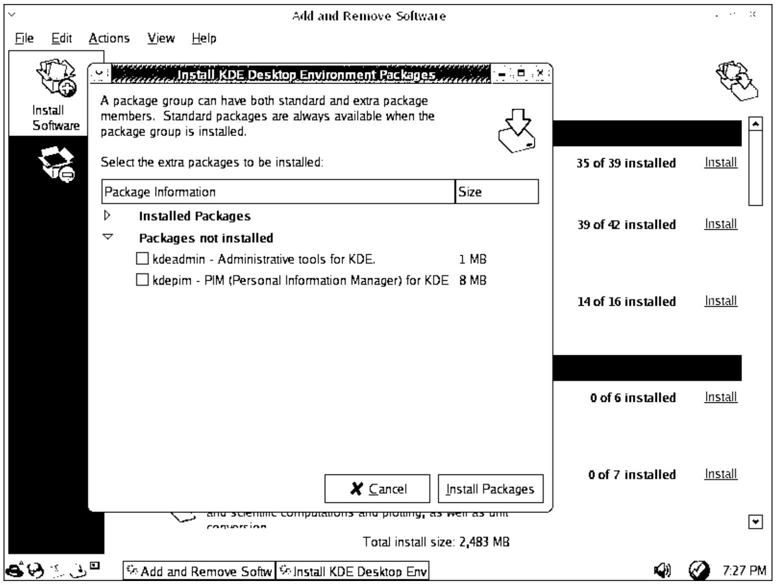
FIGURE 8.2 The initial screen of the package management tool will look familiar if you installed Fedora Core; it's the package selection screen used by the installation program.

The packages listed in the screen are organized into the groups Desktops, Applications, Servers, Development, and Systems groups. Graphical buttons on the left allow you to choose one of two modes: install or remove. To choose individual package groups for installation or removal, first click the appropriate graphical button. The numbers to the right of the package group name indicate the number of packages installed on your system and the total number of packages available in the group. In Figure 8.2, you can see that 39 of 42 possible GNOME packages have been installed.

TIP

The View menu choice allows you to toggle between the default group listing and a listing by individual packages—handy if you know the name of the package you seek. In Package View mode, check the box next to the package name to install or remove it; the package manager will always resolve any dependency issues.

Clicking on the Remove Software or Install Software buttons on the left of the Add Package Groups window brings up a window with a listing of packages associated with the related category. Figure 8.3 shows the details available for the KDE Desktop Environment package.



A package group can have both standard and extra package members. Standard packages are always available when the package group is installed.

Select the extra packages to be installed:

Package Information	Size
Installed Packages	
Packages not installed	
☐ kdeadmin - Administrative tools for KDE.	1 MB
☐ kdepim - PIM (Personal Information Manager) for KDE	8 MB

35 of 39 installed

39 of 42 installed

14 of 16 installed

0 of 6 installed

0 of 7 installed

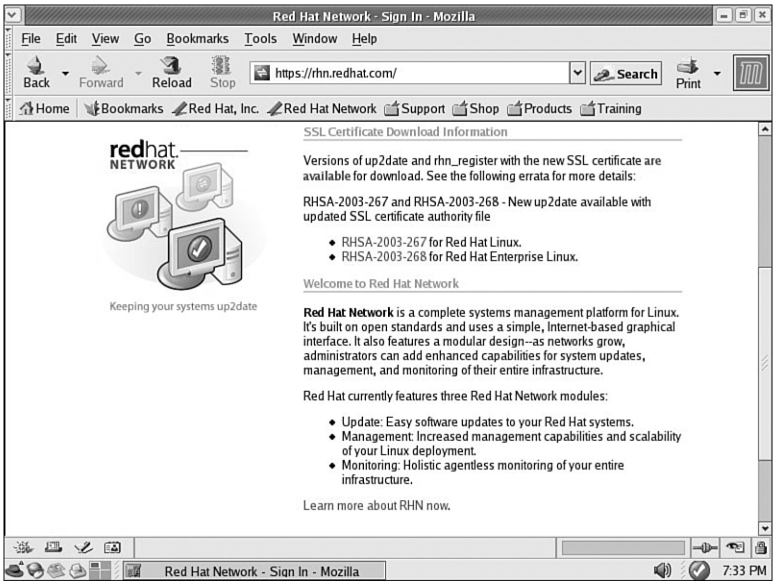
Total install size: 2,483 MB

FIGURE 8.3 The Installed Packages listing (collapsed in this view) details the base packages installed when the main category is selected. Packages Not Installed are those that you have the option of installing.

Using Red Hat Network and Alternatives for Software Management

For the Red Hat 8 and 9 releases, the only command-line application for the management of system software provided by Red Hat was RPM; the only GUI tool for management was the Red Hat Packages graphical client (`system-config-packages`) now identified as Add/Remove Software. For the average person, `RPM` is hopelessly complicated; it includes extra complexity to allow it to be used as a general `.rpm` file-building and development tool, features not normally used by many people. The graphical software management client has been improved for Fedora Core to enable the use of local and remote YUM (that stands for YellowDog Updater modified, first used in the YellowDog [PPC] Linux distribution) repositories.

Not previously covered in this book is the Red Hat Network (`https://rhn.redhat.com`—see Figure 8.4). RHN hasn't been covered because it is not free to the users of this book (or to users of the freely-available FTP download version) and it is limited to managing only the software that is provided by Red Hat. Other alternatives are now available.


Red Hat Network - Sign In - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
https://rhn.redhat.com/
Search
Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products Training
redhat. NETWORK
Keeping your systems up2date
Red Hat Network - Sign In - Mozilla
7:33 PM

SSL Certificate Download Information

Versions of up2date and rhn_register with the new SSL certificate are available for download. See the following errata for more details:

RHSA-2003-267 and RHSA-2003-268 - New up2date available with updated SSL certificate authority file

- RHSA-2003-267 for Red Hat Linux.
- RHSA-2003-268 for Red Hat Enterprise Linux.

Welcome to Red Hat Network

Red Hat Network is a complete systems management platform for Linux. It's built on open standards and uses a simple, Internet-based graphical interface. It also features a modular design--as networks grow, administrators can add enhanced capabilities for system updates, management, and monitoring of their entire infrastructure.

Red Hat currently features three Red Hat Network modules:

- Update: Easy software updates to your Red Hat systems.
- Management: Increased management capabilities and scalability of your Linux deployment.
- Monitoring: Holistic agentless monitoring of your entire infrastructure.

Learn more about RHN now.

FIGURE 8.4 The Red Hat Network provides automated update subscription services. It only works with software provided by Red Hat and is designed for commercial users.

The RHN service is a client-server mechanism through which a user subscribes to a channel on the server (a repository of software for a specific release of Fedora Core). The local client (identified by a round icon at the right of the desktop panel) can manually or automatically connect to the Red Hat Network server, obtain a list of updates, errata, and security fixes and install them; one subscription is required for each client.

NOTE

If this client-server model appeals to you and you want to use it for your network, you can sign up for subscriptions or consider one of two Open Source Up2Date server alternatives: Current at `http://current.tigris.org/`, or NRH-up2date at `http://www.nrh-up2date.org/index.html`. These two project are attempting to provide RHN-like services under you control so that you can use non-Red Hat and non-Fedora packages with the service.

Although there are several alternatives for `rpm/system-config-packages`, two of the most useful are APT and YUM. These package management applications go a long way toward solving dependency problems and easing the use of RPM to manage software. Both can install software from either local or remote repositories. Interestingly, the YUM application is now provided in Fedora, and `system-config-packages` has been modified to be used as a graphical front end for it.

The benefit to installing and using either of these two applications is that they allow you easy access to and installation of programs that Red Hat nor Fedora Core can't or won't provide (such as multimedia and non-GPL licensed applications). Since the APT and YUM mirrors for Fedora all have current Fedora Core updates, it is not necessary for you to use `up2date` or RHN to keep your computer software current. The following sections discuss the APT and YUM applications in more detail.

APT

Originally developed for Debian Linux and modified to use with `rpm` packages by Connective Linux, APT and its GUI interface Synaptic are easy to install and use. There are two primary providers of APT packages and repositories; either provider is a good choice. However, because of version conflicts between packages, the two projects' files should not be used together. FreshRPMs at `http://www.freshrpms.net/` is one provider; the Fedora Project at `http://fedora.mplug.org/` is the other. The Fedora Project has now been integrated into the briefly-lived Red Hat Linux Project and renamed Fedora Core. We will use FreshRPMs as an example because they provide packages not available from Fedora Core.

Here's how you install and configure APT:

- Read the introduction to APT in the `/apt/` section of the `http://freshrpms.net` site.
- Install the FreshRPMS GPG key (you need to have `lynx` installed, or simply download and `gpg --import` the downloaded text file):

  ```
  # lynx -source http://freshrpms.net/packages/RPM-GPG-KEY.txt ¦ gpg --import
  ```

- Install the correct version of `apt` from FreshRpms.

```
# rpm -ivh http://ftp.freshrpms.net/pub/freshrpms/fedora/linux/2/
➥APT/apt-0.5.15cnc6-1.1.fc2.fr.i386.rpm
```

- Review the `man` page of `apt`. The most important commands are `apt-get update` and `apt-get install` *`packagename`*.

  ```
  # man apt
  ```

- The list of apt repositories is preconfigured in `/etc/apt/sources/list`, but the index of packages that is available needs to be updated with

  ```
  # apt-get update
  ```

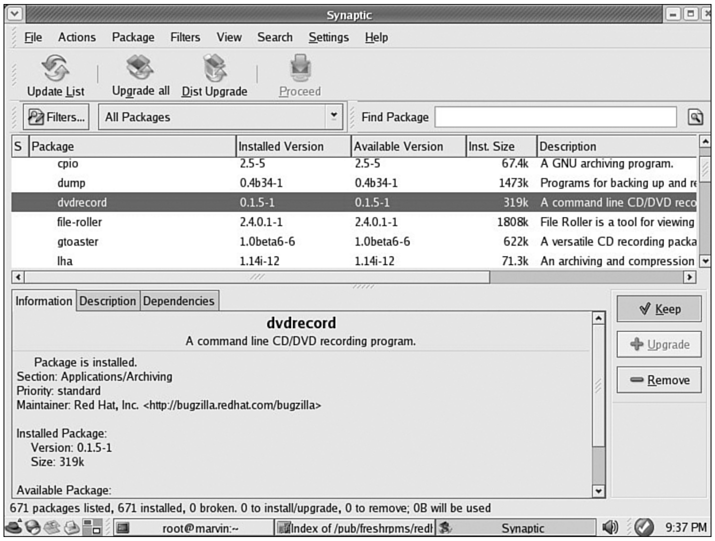
- Once the update of the packages list is completed, we suggest that you install Synaptic, the GUI package manager shown in Figure 8.5.

  ```
  # apt-get install synaptic
  ```

- Launch synaptic from the command line, browse the available packages, and install what you like.

  ```
  # synaptic &
  ```

The Synaptic graphical interface is nicely laid out (see Figure 8.5). It allows you to view a list of all available packages and any dependencies required for them. It also provides a graphical interface to add new repositories.



S	Package	Installed Version	Available Version	Inst. Size	Description
	cpio	2.5-5	2.5-5	67.4k	A GNU archiving program.
	dump	0.4b34-1	0.4b34-1	1473k	Programs for backing up and re
	dvdrecord	0.1.5-1	0.1.5-1	319k	A command line CD/DVD reco
	file-roller	2.4.0.1-1	2.4.0.1-1	1808k	File Roller is a tool for viewing
	gtoaster	1.0beta6-6	1.0beta6-6	622k	A versatile CD recording packa
	lha	1.14i-12	1.14i-12	71.3k	An archiving and compression

dvdrecord
A command line CD/DVD recording program.

Package is installed.
Section: Applications/Archiving
Priority: standard
Maintainer: Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>

Installed Package:
Version: 0.1.5-1
Size: 319k

Available Package:

671 packages listed, 671 installed, 0 broken. 0 to install/upgrade, 0 to remove; 0B will be used

FIGURE 8.5 Synaptic is a graphical interface to the APT package management application making it incredibly easy to use.

YUM

Some developers believe that although APT is a good tool, using it for `.rpm` packages is a hack. APT also is believed to be bloated with unnecessary code used for the Debian `.deb` packages. A new tool, YUM, was developed using the Python language because the Fedora Core installer, Anaconda, was written in Python and much of the code could be shared. This decision is what has made YUM the choice for integration into the Fedora Core distribution. It works much the same as APT, but lacked a GUI tool. The Fedora developers have integrated support for YUM into the graphical Red Hat/Fedora package management tool. You can obtain YUM from the Fedora site as well as from FreshRPMs.net; the home page of YUM is at `http://linux.duke.edu/projects/yum/`.

Here's how you install and configure YUM:

- Read the `HOWTO` and `Users FAQ` at the Fedora site `http://fedora.mplug.org/`.
- Install the Fedora GPG key:

  ```
  # lynx -source http://fedora.mplug.org/FEDORA-GPG-KEY ¦ gpg --import
  ```

- Install the correct version of YUM.

  ```
  # rpm -ivh http://ftp.freshrpms.net/pub/freshrpms/fedora/linux/2
  /yum/yum-2.0.7-2.fc.fr.noarch.rpm
  ```

- Review the `man` page for `yum` to familiarize yourself with all available options.

  ```
  # man yum
  ```

Once YUM is installed, the following commands are useful (remember that the Fedora graphical client is available if you configure it to use YUM repositories):

`yum list`—A list of all packages available from the repository.

`yum list installed`—A list of all packages installed on your computer.

`yum list updates`—A list of all updates available for your computer.

`yum install` *packagename*—Installs *packagename*.

`yum update`—Run without a *packagename*, YUM will update all installed packages.

`yum remove` *packagename*—Removes a package and dependencies.

`yum upgrade`—Run without a package name, YUM will upgrade all packages and remove any obsoleted packages; `yum update` will not remove obsoleted packages.

You'll find either APT or YUM to be useful additions to Fedora Core. We suggest that you use the one from FreshRPMs.net if you want to install some of the multimedia applications described in Chapter 26, "Multimedia Applications."

Compiling Software from Source

Not all the software you might want to use is available in rpm packages or in the exact form that you desire. Many complicated programs have options that can only be selected at compile time and many smaller, special-purpose applications only exist as source code. Fedora Core provides all the tools necessary for you to compile source code into binary applications. First, we'll cover building from source rpm files, and then manipulating source rpms files and finally, building from source tarballs.

NOTE

For other software package formats, you can use the File Roller application (found in the Accessories menu) to easily display, browse, read, and extract contents of compressed files, including legacy archives such as tarballs or compressed tar archives (recognized by their .gz or .tgz extensions). Other compressed files, such as those created with the compress command (ending in .Z) or bzip2 files (ending in .bz2), are also supported by File Roller. The File Roller client will also convert compressed archives between gzip and bzip2 formats.

Building RPMS from src.rpm Files

A rule of thumb is that you never build rpms as the root user even though the directories are already set up at /usr/src/redhat as follows:

```
# tree /usr/src/redhat
/usr/src/redhat
¦-- BUILD
¦-- RPMS
¦   ¦-- athlon
¦   ¦-- i386
¦   ¦-- i486
¦   ¦-- i586
¦   ¦-- i686
¦   `-- noarch
¦-- SOURCES
¦-- SPECS
`-- SRPMS
```

Using the mkdir command, re-create this directory tree structure in your home directory; you can name the new directory redhat (or anything you like). You might even want to create a new user just to build rpms and source code. (You can compile without being root; you just can't install the software system-wide as a regular user.)

The configuration information for building `rpms` is kept in three places:

`/usr/lib/rpm/*/macros`—The systemwide defaults.

`/etc/rpm/macros.*`—Where systemwide changes are kept.

`~/.rpmmacros`—Where user-specific changes are kept.

Because we need to tell `rpm` that we will not be using the systemwide default build location for our user, we can

```
$ echo "%_topdir $HOME/redhat" > $HOME/.rpmmacros
```

TIP

Here, we use > instead of >> to blank the file in case there is already content in it. The >> construct appends information to a file.

To select a temporary directory

```
$ echo "%_tmppath $HOME/tmp" >> $HOME/.rpmmacros
```

To set any compiler optimization flags (here, we're using an Athlon processor as an example), we'll use

```
$ echo "-o3 -march=athlon" >> $HOME/.rpmmacros
```

To rebuild a `src.rpm` file as a regular user, you would use

```
$ rpmbuild --recompile packagename.src.rpm
```

After a successful build, you will find the binary file(s) in `~/redhat/RPMS/athlon`.

You can install them as `root` with

```
# rpm -Uvh --replacepkgs --replacefiles packagename.rpm
```

If the build fails, the error message will point you to a solution (usually a dependency has not been satisfied). You'll find that a lot of the packages named with `-devel` will be needed before you compile from source code. Install the package for the missing dependency and retry the compile.

Working with Source RPM Files

You might want to modify a source package for your own uses such as adding documentation, changing default settings, or patching the source code itself. Fedora Core provides the source code to its distribution in the form of source RPM files. You can access the source code on disks 4 and 5 of the downloadable CD images or obtain them from the Fedora Core FTP site.

TIP

An important part of the RPM file is called the `.spec` file, a specification file. It tells RPM how to behave with the particular source files during compilation, installation, and erasure.

As an example, we'll use information that was found at `http://elektron.its.tudelft.nl/~rbos36/mdkfreetype2.html` (the page has now been removed by the author) to modify the freetype2 library provided with Fedora Core in order to enable the bytecode interpreter. The code for the interpreter has been disabled by default because of redistribution licensing concerns that don't affect individual use. Enabling the interpreter will result in improved rendering of the TrueType fonts. We used the file from Red Hat 7.3 as our example, but the source file from 10 should work as well.

Begin work by first installing the source RPM package with `rpm -i`. (Note that here we are building as root to follow the example from the Web page; you should typically build packages as a regular user.) In our example, obtain the `freetype-2.0.9-2.src.rpm` and install it with `rpm -i`. The source code files are placed in `/usr/src/redhat/SOURCES`.

Copy the source file (it's a compressed `tar` file) to `/tmp`, and then `cd` (change directories) there to unpack and modify it:

```
# cp freetype-2.0.9.tar.bz2 /tmp
# cd /tmp
```

Because it's a `.bz2` (BZip2 compressed) `tar` file, un-tar it with

```
# tar xjvf freetype-2.0.9.tar.bz2
```

and `cd` to the new directory:

```
# cd freetype-2.0.9
```

Using the text editor of your choice, edit the file `include/freetype/config/ftoption.h` and find the line

```
#undef TT_CONFIG_OPTION_BYTECODE_INTERPRETER
```

Change it to

```
#define TT_CONFIG_OPTION_BYTECODE_INTERPRETER
```

Save it and exit the text editor.

Next, re-create the compressed archive:

```
# cd /tmp
# tar cfj freetype-2.0.9.tar.bz2 ./freetype-2.0.9/
```

Put it back in your source directory:

```
# mv freetype-2.0.9.tar.bz2 /usr/src/SOURCES
```

Now edit the `.spec` file in `/usr/src/redhat/SPECS` to change the line beginning with `Release` to increment the number found there. (We are changing the version number by doing this, so it will not conflict with the version of the application we will be replacing.) Make any changes to the `%description` line to describe your changes if you desire, and save the file.

Build the binary RPM with

```
# rpmbuild -bb freetype.spec
```

During the build process, RPM will detect a patch and ask you about the patch; press `y` for "yes" to continue.

The new RPMs (actually four of them) are found in `/usr/src/redhat/RPMS/i386`. We only need the one named `freetype-2.0.9`; you can install it with `rpm -Uvh`. (This is why we changed the version number; if we had not, RPM would not upgrade to the "same" version. Had we not changed the version number, we could have forced the installation with the `--replacepackages --replacefiles` option.)

The font server needs to be restarted to use the new library, so we use the `service` command as shown in Chapter 7, "Managing Services."

```
# service xfs restart
```

Enjoy your new look—provided by better rendering of the fonts.

Compile from Source Tarballs

Compiling applications from source is not that difficult. Most source code is available as compressed source *tarballs*—that is, `tar` files that have been compressed using `gzip` or `bzip`. The compressed files will typically uncompress into a directory containing several files. It's always a good idea to compile source code as a regular user to limit any damage that broken or malicious code might inflict, so create a directory named `source` in your home directory.

From wherever you downloaded the source tarball, uncompress it into the `~/source` directory using the `-C` option to `tar`:

```
$ tar zxvf packagename.tgz -C  ~/source

$ tar zxvf packagename.tar.gz -C  ~/source

$ tar jxvf packagename.bz -C  ~/source

$ tar jxvf packagename.tar.bz2 -C  ~/source
```

If you're not certain what file compression method was used, employ the `file` command to figure it out:

```
$ file packagename
```

Now, change directories to `~/source/`*`packagename`* and look for a file named `README`, `INSTALL`, or a similar name. Print the file out if necessary because it will contain specific instructions on how to compile and install the software. Typically, the procedure to compile source code is

```
$ ./configure
```

which runs a script to check if all dependencies are met and the build environment is correct. Then,

```
$ make
```

to compile the software. And finally, as root:

```
# make install
```

If the compile fails, check the error messages for the reason and run

```
$ make clean
```

before you start again. You can also run

```
$ make uninstall
```

to remove the software if you don't like it.

An alternative to running `make install` is a program named CheckInstall, which will produce an `rpm` file for your installation. This method allows the RPM database to be aware of and keep track of all the files you are installing. See the following sidebar on CheckInstall for more information.

A HANDY SOFTWARE INSTALLATION TOOL—CHECKINSTALL

When you compile applications from source and install them, they won't show up in the RPM database and therefore can't be managed by RPM.

You can provide RPM management for these applications by using a program named CheckInstall. At its simplest, `checkinstall` is a drop-in substitute for the `make install` step in building from source code.

For example, when compiling from source code, you would traditionally use

```
# ./configure
# make
# make install
```

Using CheckInstall, the steps would look like this:

```
# ./configure
# make
# checkinstall
```

CheckInstall will create a binary `.rpm` package and install the application from it. This makes the new application part of the RPM database. The new `.rpm` file is left for you in `/usr/src/redhat/RPMS/i386`. The new application can later be uninstalled with `rpm -e`.

Some applications arrive in the form of a shell script wrapper. Using the shell script as the argument to CheckInstall will provide a `.rpm` file of the installed files and add them to your RPM database. Not all applications will install with CheckInstall. Read its accompanying documentation carefully.

CheckInstall can be downloaded from `http://asic-linux.com.mx/~izto/checkinstall/index.php`.

System Monitoring Tools

Monitoring your server or workstation is an important task, especially in a commercial or corporate environment. Whether you're working on critical application programming or conducting e-commerce on the Internet, you'll want to track your system's health signs while it's running. Good Fedora Core Linux system administrators are also quite vigilant about watching running processes on their systems, including resources such as CPU and disk, memory, network, and printer usage. Even though the task isn't strictly part of standard security operations, such as examining system logs and network traffic, monitoring resource usage can help you spot misuse and avoid developing problems, such as unwanted intruder connections to your network.

The next sections introduce just a few of the basic tools and approaches used to monitor a running Linux system. Some of the tools focus on in-memory processes, whereas others, such as filesystem reporting and network monitoring, have more comprehensive uses. You'll also see how to control some system processes using various command-line and graphical tools included with Fedora Core Linux.

Console-Based Monitoring

Traditional UNIX systems have always included the `ps` or process display command. This command lists the running processes on the system and identifies who owns them and how much of the system resources are being used.

Because of the architecture of the Linux kernel and its memory management, Linux also provides much process reporting and control via the command line. This feature can be accessed manually through the `/proc` filesystem, a pseudo-filesystem used as a direct interface to the kernel. (You see how it's used in the upcoming discussion of the `ps` command.)

The /proc filesystem is frequently used by application programmers who construct an interface for the raw information it provides. This filesystem is too complex to adequately deal with in the context of this chapter, but you can benefit from reading the proc man page and /usr/src/linux-2.4/Documentation/filesystems/proc.txt to examine the list and description of the scores of kernel values available. You then can write shell scripts (see Chapter 22, "Shell Scripting") to use those values as needed.

Processes can be controlled at the command line as well. Whenever a program or command is launched on your Fedora Core Linux system, the process started by the kernel is assigned an identification number, called a PID or Process ID. This number is (generally) displayed by the shell if the program is launched in the background, like this:

```
$ xosview &
[1] 11670
```

In this example, the xosview client has been launched in the background, and the (bash) shell reported a shell job number ([1] in this case). A job number or job control is a shell-specific feature that allows a different form of process control (such as sending or suspending programs to the background and retrieving background jobs to the foreground; see your shell's manual pages for more information if you are not using bash).

The second number displayed (11670, in this example) represents the Process ID. You can get a quick list of your processes by using the ps command like this:

```
$ ps
  PID TTY          TIME CMD
  736 tty1     00:00:00 bash
  743 tty1     00:00:00 startx
  744 tty1     00:00:00 tee
  752 tty1     00:00:00 xinit
  756 tty1     00:00:09 kwm
  ...
11670 pts/4    00:00:00 xosview
11671 pts/4    00:00:00 ps
```

Note that not all output from the display is shown here. But as you can see, the output includes the process ID, abbreviated as PID, along with other information, such as the name of the running program. Like any UNIX command, many options are available; the proc man page has a full list. A most useful option is aux, which provides a friendly list of all the processes. You should also know that ps works not by polling memory, but through the interrogation of the Linux /proc or process filesystem. (ps is one of the interfaces mentioned at the beginning of this section.)

The /proc directory contains quite a few files—some of which include constantly updated hardware information (such as battery power levels, and so on). Linux administrators will often pipe the output of ps through a member of the grep family of commands in order to display information about a specific program, perhaps like this:

```
$ ps aux ¦ grep xosview
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
bball          11670  0.3    1.1      2940  1412 pts/4
➥S    14:04        0:00    xosview
```

This example returns the owner (the user who launched the program) and the PID, along with other information, such as the percentage of CPU and memory usage, size of the command (code, data, and stack), time (or date) the command was launched, and name of the command. Processes can also be queried by PID like this:

```
$ ps 11670
  PID TTY      STAT   TIME COMMAND
11670 pts/4    S      0:00 xosview
```

You can use the PID to stop a running process by using the shell's built-in `kill` command. This command will ask the kernel to stop a running process and reclaim system memory. For example, to stop the `xosview` client in the example, use the `kill` command like this:

```
$ kill 11670
```

After you press Enter (or perhaps press Enter again), the shell might report

```
 [1]+  Terminated              xosview
```

Note that users can only `kill` their own processes, but `root` can kill them all. Controlling any other running process requires root permission, which should be used judiciously (especially when forcing a `kill` by using the `-9` option); by inadvertently killing the wrong process through a typo in the command, you could bring down an active system.

Using the `kill` Command to Control Processes

The `kill` command is a basic UNIX system command. We can communicate with a running process by entering a command into its interface, such as when we type into a text editor. But some processes (usually system processes rather than application processes) run without such an interface, and we need a way to communicate with them as well, so we use a system of signals. The `kill` system accomplishes just that by sending a signal to a process, and we can use it to communicate with any process. The general format of the `kill` command is

```
# kill option PID
```

A number of signal options can be sent as words or numbers, but most are of interest only to programmers. The most common ones you will use are

```
# kill PID
```

This tells the process with that PID to stop. (You supply the actual PID.)

```
# kill -9 PID
```

This is the signal for `kill` (`9` is the number of the SIGKILL signal); use this combination when the plain `kill` shown previously doesn't work.

```
# kill -SIGHUP PID
```

This is the signal to "hangup"—stop—and then clean up all associated processes as well. (Its number is `-1`.)

As you become proficient at process control and job control, you will learn the utility of a number of `kill` options. A full list of signal options can be found in the `man signal` page.

Using Priority Scheduling and Control

Every process cannot make use of the systems resources (CPU, memory, disk access, and so on) as it pleases. After all, the kernel's primary function is to manage the system resources equitably. It does this by assigning a priority to each process so that some processes get better access to system resources and some processes might have to wait longer until their turn arrives. Priority scheduling can be an important tool in managing a system supporting critical applications or in a situation in which CPU and RAM usage must be reserved or allocated for a specific task. Two legacy applications included with Red Linux include the `nice` and `renice` commands. (`nice` is part of the GNU `sh-utils` package, whereas `renice` is inherited from BSD UNIX.)

The `nice` command is used with its `-n` option, along with an argument in the range of `-20` to `19`, in order from highest to lowest priority (the lower the number, the higher the priority). For example, to run the `xosview` client with a low priority, use the `nice` command like this:

```
$ nice -n 12 xosview &
```

The `nice` command is typically used for disk or CPU-intensive tasks that might be obtrusive or cause system slowdown. The `renice` command can be used to reset the priority of running processes or control the priority and scheduling of all processes owned by a user. Regular users can only numerically increase process priorities (for example, make tasks less important) using this command, but the root operator can use the full `nice` range of scheduling (-20 to 19).

System administrators can also use the `time` command (here, `time` is used to measure the duration of elapsed time; the command that deals with civil and sidereal time is the `date` command) to get an idea about how long and how much of a system's resources will be required for a task (such as a shell script). This command is used with the name of another command (or script) as an argument like this:

```
# time -p find / -name core -print
/dev/core
/proc/sys/net/core
```

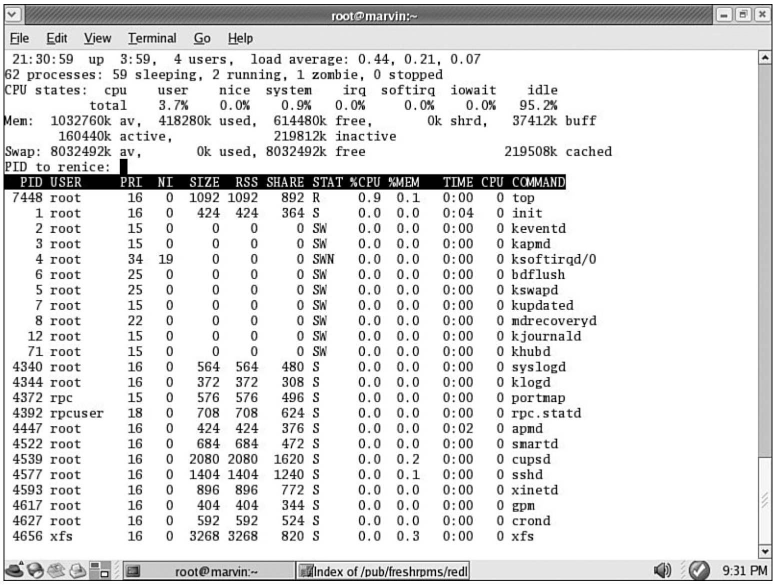
```
real 1.20
user 0.14
sys 0.71
```

Output of the command displays the time from start to finish, along with user and system time required. Other factors you can query include memory, CPU usage, and filesystem Input/Output (I/O) statistics. See the `time` command's manual page for more details.

Nearly all graphical process monitoring tools include some form of process control or management. Many of the early tools ported to Linux were clones of legacy UNIX utilities. One familiar monitoring (and control) program is `top`. Based on the `ps` command, the `top` command provides a text-based display, constantly updated console-based output showing the most CPU-intensive processes currently running. It can be started like this:

```
# top
```

After you press Enter, you'll see a display as shown in Figure 8.6. The `top` command has a few interactive commands: pressing `h` displays the help screen; pressing k prompts you to enter the `pid` of a process to kill; pressing `n` prompts you to enter the `pid` of a process to change its nice value. The `top` man page describes other commands and includes a detailed description of what all the columns of information `top` can display actually represent; have a look at top's well-written `man` page.



```
 21:30:59  up  3:59,  4 users,  load average: 0.44, 0.21, 0.07
62 processes: 59 sleeping, 2 running, 1 zombie, 0 stopped
CPU states:  cpu    user    nice  system    irq  softirq  iowait    idle
           total    3.7%    0.0%    0.9%   0.0%     0.0%    0.0%   95.2%
Mem:  1032760k av,  418280k used,  614480k free,       0k shrd,   37412k buff
       160440k active,             219812k inactive
Swap: 8032492k av,       0k used, 8032492k free                  219508k cached
PID to renice: 
  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
 7448 root      16   0  1092 1092   892 R     0.9  0.1   0:00   0 top
    1 root      16   0   424  424   364 S     0.0  0.0   0:04   0 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 keventd
    3 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kapmd
    4 root      34  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd/0
    6 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    5 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 kswapd
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kupdated
    8 root      22   0     0    0     0 SW    0.0  0.0   0:00   0 mdrecoveryd
   12 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kjournald
   71 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 khubd
 4340 root      16   0   564  564   480 S     0.0  0.0   0:00   0 syslogd
 4344 root      16   0   372  372   308 S     0.0  0.0   0:00   0 klogd
 4372 rpc       15   0   576  576   496 S     0.0  0.0   0:00   0 portmap
 4392 rpcuser   18   0   708  708   624 S     0.0  0.0   0:00   0 rpc.statd
 4447 root      16   0   424  424   376 S     0.0  0.0   0:02   0 apmd
 4522 root      16   0   684  684   472 S     0.0  0.0   0:00   0 smartd
 4539 root      16   0  2080 2080  1620 S     0.0  0.2   0:00   0 cupsd
 4577 root      16   0  1404 1404  1240 S     0.0  0.1   0:00   0 sshd
 4593 root      16   0   896  896   772 S     0.0  0.0   0:00   0 xinetd
 4617 root      16   0   404  404   344 S     0.0  0.0   0:00   0 gpm
 4627 root      16   0   592  592   524 S     0.0  0.0   0:00   0 crond
 4656 xfs       16   0  3268 3268   820 S     0.0  0.3   0:00   0 xfs
```

FIGURE 8.6 The `top` command can be used to monitor and control processes. Here, we are being prompted to re-`nice` a process.

The top command displays quite a bit of information about your system. Processes can be sorted by PID, age, CPU or memory usage, time, or user. This command also provides process management, and system administrators can use its k or r keypress commands to kill or reschedule running tasks.

The top command uses a fair amount of memory, so you might want to be judicious in its use and not leave it running all the time.

Displaying Free and Used Memory with free

Although top includes some memory information, the free utility will display the amount of free and used memory in the system in kilobytes (the -m switch displays in megabytes). On one system, the output looks like this:

```
# free
                   total     used      free     shared   buffers   cached
Mem                255452    251132    4320     0        19688     77548
-/+ buffers/cache: 153896    101556
Swap:              136512    31528     104984
```

This output describes a machine with 256MB of RAM memory and a swap partition of 137MB. Note that some swap is being used although the machine is not heavily loaded. Linux is very good at memory management and "grabs" all the memory it can in anticipation of future work.

TIP

A useful trick is to employ the watch command; it will repeatedly rerun a command every 2 seconds by default. If you use

```
#  watch free
```

you'll see the output of the free command updated every two seconds.

Another useful system monitoring tool is vmstat (virtual memory statistics). This command reports on processes, memory, I/O, and CPU typically providing an average since the last reboot, or you can make it report usage for a current period of time by telling it the time interval in seconds and the number of iterations you desire, like

```
# vmstat 5 10
```

which will run vmstat every five seconds for ten iterations.

Use the uptime command to see how long it has been since the last reboot and to get an idea of what the load average has been; higher numbers mean higher loads.

Disk Quotas

Disk quotas are a way to restrict the usage of disk space either by user or by groups. Although rarely—if ever—used on a local or standalone workstation, quotas are definitely a way of life at the enterprise level of computing. Usage limits on disk space not only conserve resources, but also provide a measure of operational safety by limiting the amount of disk space any user can consume.

Disk quotas are more fully covered in Chapter 9, "Managing Users."

Graphical Process and System Management Tools

The GNOME and KDE desktop environments offer a rich set of network and system monitoring tools. Graphical interface elements, such as menus and buttons, and graphical output, including metering and real-time load charts, make these tools easy to use. These clients, which require an active X session and in some cases (but not all) root permission, are included with Fedora Core Linux.

If you view the graphical tools locally while they are being run on a server, you must have X properly installed and configured on your local machine. Although some tools can be used to remotely monitor systems or locally mounted remote filesystems, you'll need to properly configure pertinent X11 environment variables, such as `$DISPLAY`, in order to use the software or use the `ssh` client's `-X` option when connecting to the remote host.

Fedora Core Linux includes the `xosview` client, which provides load, CPU, memory and swap usage, disk I/O usage and activity, page swapping information, network activity, I/O activity, I/O rates, serial port status, and if APM is enabled, the battery level (such as for a laptop).

For example, to see most of these options, start the client like this:

```
# xosview -geometry 406x488 -font 8x16 +load +cpu +mem +swap \
 +page +disk +int +net &
```

After you press Enter, you'll see a display as shown in Figure 8.7.

The display can be customized for a variety of hardware and information, and the `xosview` client (like most well-behaved X clients) obeys geometry settings such as size, placement, or font. If you have similar monitoring requirements, but want to try a similar but different client from `xosview`, try `xcpustate`, which has features that enable it to monitor network CPU statistics foreign to Linux. Neither of these applications is installed with the base set of packages; you need to install them manually if you want to use them.

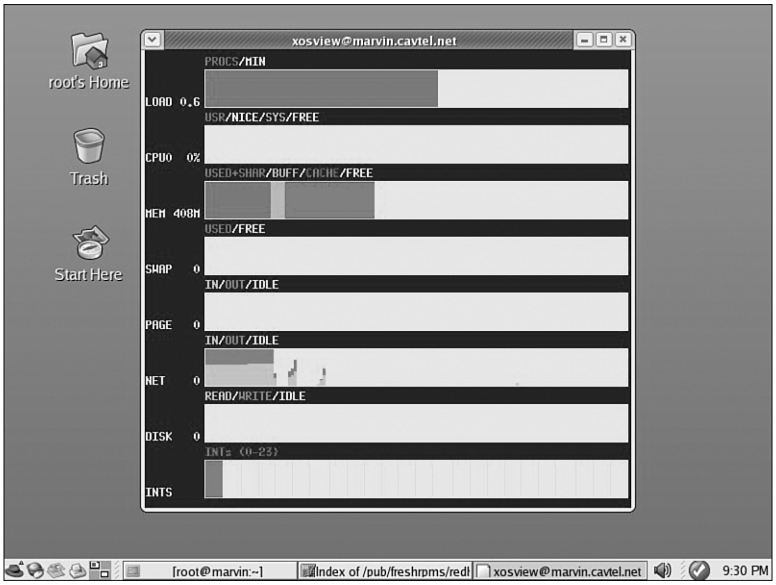
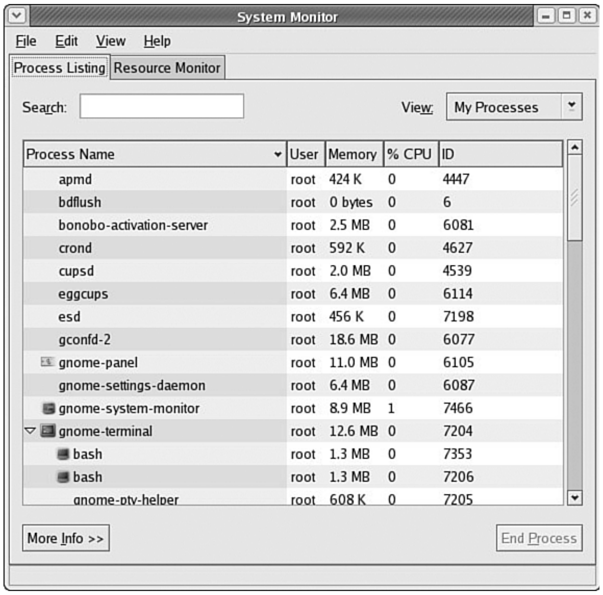


FIGURE 8.7 The `xosview` client displays basic system stats in a small window. You can choose from several options to determine what it will monitor for you.

Some of the graphical system and process monitoring tools included with Fedora Core Linux include the following:

- `vncviewer`—AT&T's open source remote session manager (part of the Xvnc package), which can be used to view and run a remote desktop session locally. This software (discussed in more detail in Chapter 27, "Using Emulators and Cross-Platform Tools") requires an active, but background X session on the remote computer.
- `nmapfe`—A GTK+ graphical front end to the `nmap` command. This client provides system administrators with the ability to scan networks to monitor the availability of hosts and services.
- `ethereal`—This graphical network protocol analyzer can be used to save or display packet data in real time and has intelligent filtering to recognize data signatures or patterns from a variety of hardware and data captures from third-party, data-capture programs, including compressed files. Some protocols include AppleTalk, Andrew File System (AFS), AOL's Instant Messenger, various Cisco protocols, and many more.
- `gnome-system-monitor`—Replacing `gtop`, this tool is a simple process monitor offering two views: a list view and a moving graph. It is accessed via the System Tool menu selection as the System Monitor item (see Figure 8.8).



View: My Processes

Process Name	User	Memory	% CPU	ID
apmd	root	424 K	0	4447
bdflush	root	0 bytes	0	6
bonobo-activation-server	root	2.5 MB	0	6081
crond	root	592 K	0	4627
cupsd	root	2.0 MB	0	4539
eggcups	root	6.4 MB	0	6114
esd	root	456 K	0	7198
gconfd-2	root	18.6 MB	0	6077
gnome-panel	root	11.0 MB	0	6105
gnome-settings-daemon	root	6.4 MB	0	6087
gnome-system-monitor	root	8.9 MB	1	7466
gnome-terminal	root	12.6 MB	0	7204
bash	root	1.3 MB	0	7353
bash	root	1.3 MB	0	7206
gnome-pty-helper	root	608 K	0	7205

FIGURE 8.8 The Process Listing view of the System Monitor.

The System Monitor menu item (shown in Figure 8.8) is found in the System Tools menu. It can be launched from the command line with

```
# gnome-system-monitor
```

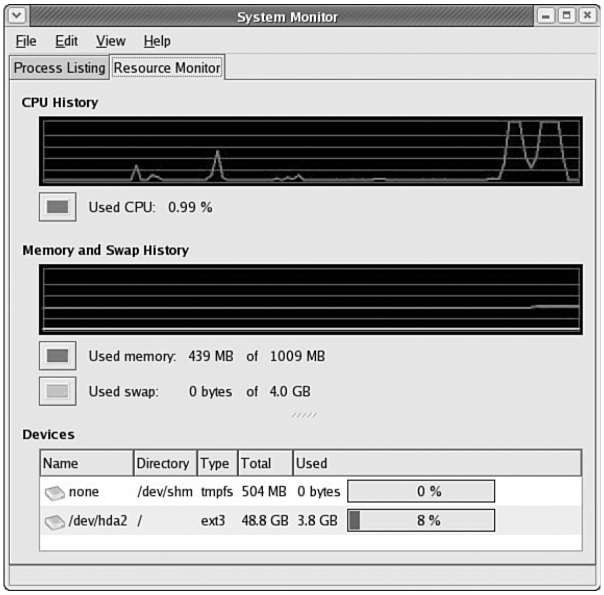
From the Process Listing view (chosen via the tab in the upper left portion of the window), select a process and click on More Info at the bottom left of the screen to display details on that process at the bottom of the display. You can select from three views to filter the display, available in the drop-down View list: All Processes, My Processes (those you alone own), or Active Processes (All Processes that are active).

Choose Hidden Processes under the Edit command accessible from the top of the display to show any hidden processes (those that the kernel does not enable the normal monitoring tools to see). Select any process and kill it with End Process.

The processes can be re-`nice`d by selecting Edit, Change Priority. The View selection from the menu bar also provides a memory map. In the Resource Monitor tab, you can view a moving graph representing CPU and memory usage (see Figure 8.9).

KDE Process and System Monitoring Tools

KDE provides several process and system monitoring clients. The KDE graphical clients are integrated into the desktop taskbar by right-clicking on the taskbar and following the menus.



Used CPU: 0.99 %

Used memory: 439 MB of 1009 MB

Used swap: 0 bytes of 4.0 GB

Name	Directory	Type	Total	Used	
none	/dev/shm	tmpfs	504 MB	0 bytes	0 %
/dev/hda2	/	ext3	48.8 GB	3.8 GB	8 %

FIGURE 8.9 The Graph view of the System Monitor. It shows CPU usage, Memory/Swap usage, and disk usage. To get this view, select the Resource Monitor tab.

These KDE monitoring clients include the following:

- `kdf`—A graphical interface to your system's filesystem table that displays free disk space and enables you to mount and unmount filesystems using a pointing device.
- `ksysguard`—Another panel applet that provides CPU load and memory use information in animated graphs.

RELEVANT FEDORA CORE AND LINUX COMMANDS

You'll use these commands while managing your Fedora Core Linux system resources and software packages:

`system-config-packages`—The Fedora Core GUI Package Manager.

`nice`—Runs a program at a specified priority.

`rpm`—The RPM Package Manager.

`ps`—Displays a list all running processes.

`top`—Displays and manages running processes.

`rpmbuild`—Builds RPM source and binary packages.

`uptime`—Tells the length of time since the last reboot and load average.

`vmstat`—Provides virtual memory statistics.

`kill`—Stops a process.

Reference

`http://www.rpm.org`—Home page for the Fedora Core Package Manager. This site provides essential links to the latest version of RPM software for Linux and X desktop environments, such as GNOME.

`http://www.rpm.org/max-rpm/`—Link to the start of an update to Ed Bailey's classic tome and RPM reference book, *Maximum RPM*.

`http://linux.tnc.edu.tw/techdoc/maximum-rpm/rpmbook/node15.html`—History of the Red Hat Package Manager.

`http://www.smoogespace.com/documents/behind_the_names.html`—A history of Red Hat Linux releases, and a good place to learn about the connection between the names of the releases.

`http://www.gnupg.org/`—Home page for GNU Privacy Guard, an unencumbered free replacement for Pretty Good Privacy.

`http://www.debian.org/doc/manuals/project-history/ch-detailed.en.html`—History of the Debian Linux package system.

`http://and.sourceforge.net/`—Home page of the `and` auto nice daemon, which can be used to prioritize and reschedule processes automatically.

`http://sourceforge.net/projects/schedutils/`—Home page for various projects offering scheduling utilities for real-time scheduling.

`http://freetype.sourceforge.net/patents.html`—A discussion of the FreeType bytecode interpreter patents.

`http://www.ethereal.com`—Home page for the Ethereal client.

`http://www.realvnc.com/`—The home page for the Virtual Network Computing remote desktop software, available for a variety of platforms, including Fedora Core Linux. This software has become so popular that it is now included with nearly every Linux distribution.

`http://www.nrh-up2date.org/index.html`—A replacement for Red Hat's Up2Date application.

`http://apt-rpm.tuxfamily.org`—An alternative to the default package manager.