O12

Red Hat Package Manager

Matilde L. Valdez

IBM eserver xSeries
Technical Conference

Aug. 9 - 13, 2004

Chicago, IL

RETURN TO INDEX

Objectives

After completing this unit, you should be able to:

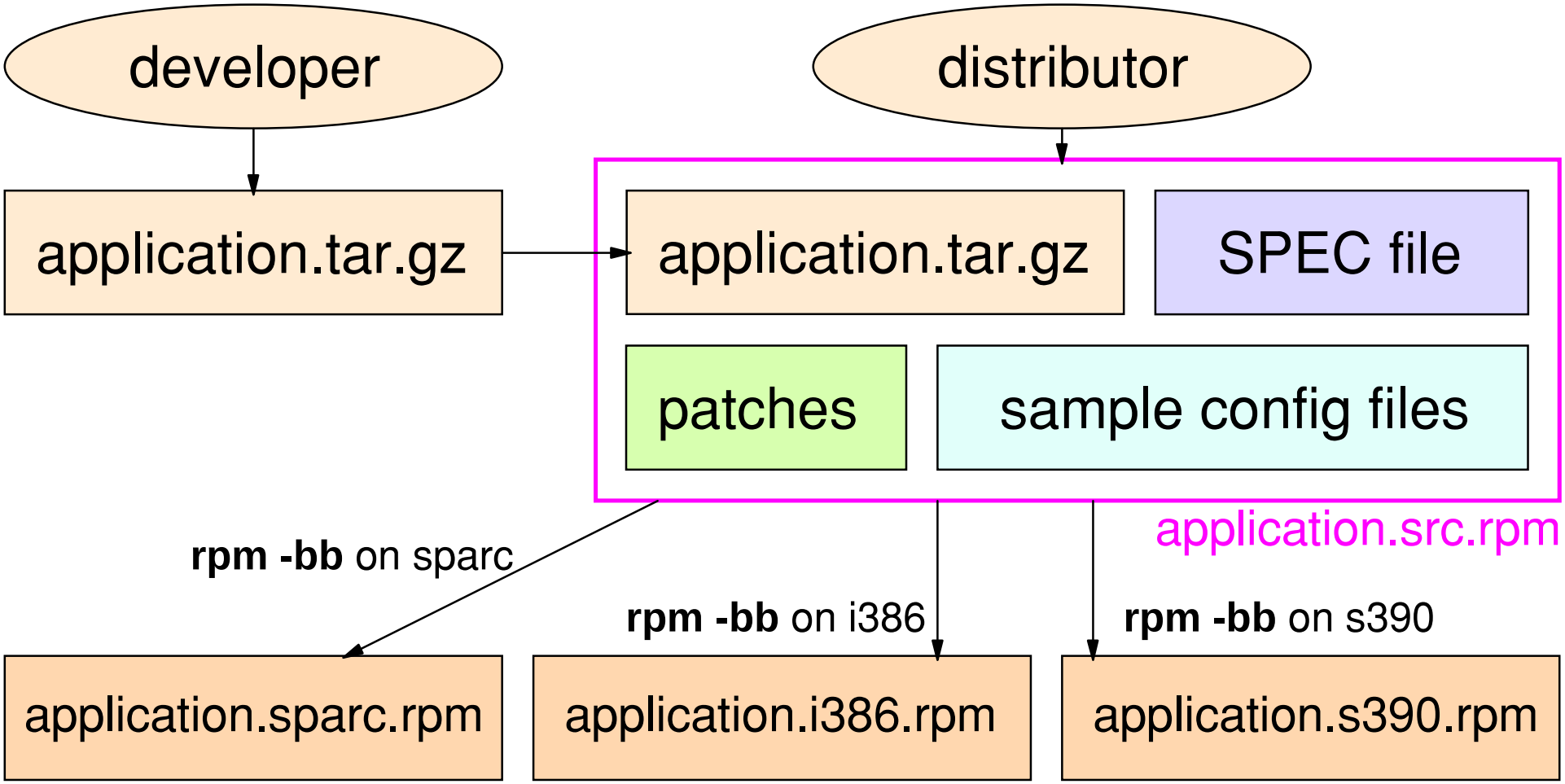
- Describe the basic principles of RPM
- Install RPM packages

RPM Package Manager (RPM)

- Used for package management
  - Management of source files
  - Build process
  - Distribution of binary files
- Developed by Red Hat Software Inc, but GPLed
  - Other Linux distributions use it too, e.g. SuSE
  - Requirement of LSB
- .rpm files can be created by distributor or others
- RPM database (/var/lib/rpm) contains database of installed packages
- Can use PGP/GPG for package signing (verification of authenticity)

Notes:

RPM Philosophy



Note: Red Hat 8.0 and up uses **rpmbuild** instead of **rpm** for building RPMs

Notes:

RPM Installing, Freshening and Upgrading

- Installs, freshens or upgrades an RPM
  - Freshen: only install if an older RPM was installed
  - Upgrade: always install, but uninstall older RPM first
- Basic syntax:

```
rpm -i package-filename.rpm   (install)
rpm -F package-filename.rpm   (freshen)
rpm -U package-filename.rpm   (upgrade)
```

- Useful options:
  -v be verbose
  -h print 50 hash marks during installation

```
# rpm -ihv package-10.2-67.i386.rpm
package: ##################################################
```

Notes:

Installing an RPM can only be done if it was not already installed. If the RPM was already installed, you need to do an upgrade or a freshen. The difference between an upgrade and a freshen is that an upgrade will always install an RPM, even when a previous version was not installed. (It will act like a regular installation in that case.) A freshen only installs packages that actually have been installed previously. A freshen therefore is very handy to use if you downloaded a lot of patches from the Red Hat site, and you are not sure which patches you actually need. You can then just freshen all the packages, and only the things you need will actually be installed.

The basic syntax for installing, freshening and upgrading is respectively:

```
rpm -i package-filename.rpm
rpm -F package-filename.rpm
rpm -U package-filename.rpm
```

Note that there is a difference between the package name and the package filename. The RPM file which contains the package foo would generally be called foo-version-release.architecture.rpm.

There are a number of options which make life a little easier on you:

- -v gives more information on what rpm is doing (verbose).
- -h prints 50 hash marks while installing, so that you can track the progress. If you run rpm from a script, you can use these hash marks to make your own progress bar.
- --nodeps disables dependency checking.

Files in an RPM are marked as program, documentation or configuration files. When doing an upgrade or freshen, program and documentation files are automatically overwritten. Configuration files are another matter altogether: Depending on the MD5 checksum of the original, actual and new configuration file, the configuration file may be left in place, may be overwritten, may be saved with an extension .rpmsave, or may be saved with an extension .rpmorig. In fact, rpm can distinguish between six different cases. For more information, see the Maximum RPM book.

When installing, freshening or upgrading packages, you may also specify the Web address of the package file instead of the package file itself. This allows you to do upgrades even on systems which are very tight on disk space, but do have access to a network (for instance the Internet). Just ensure that the RPM files can be reached, either through FTP or HTTP, and you can do an upgrade. If you need to go through a proxy, there are options available to specify this proxy as well. Look at the rpm manual page for details.

RPM Uninstalling

- For uninstalling an RPM use the -e option

```
# rpm -e kdelibs3
error: removing these packages would break dependencies:
          kdelibs3 >= 3.1 is needed by kdebase3-3.1.1-63
          libDCOP.so.4 is needed by kdelibs3-cups-3.1.1-13
         ...
```

Options:

--nodeps ignore any dependency breaks

Notes:

RPM Querying

- Queries the contents of an installed RPM
- Basic syntax:

  `-rpm -q package-name`

- Options:

-a	query all installed packages
-f <file>	query package which owns file.
-p <package-file>	query package-file
-i	display package information
-l	display package files
-s	display state of all files
-d	display documentation files
-c	display configuration files

Notes:
RPM Querying is the process of retrieving information about installed packages. The basic syntax is rpm -q package-name, but that will only display the package name. It's the options that make querying interesting:
-a queries all packages which are installed on the system.
-f <file> queries which package contains <file>.
-p <package-file> queries the (not yet installed) <package-file>.
-i displays all package information: name, version, release, install date, group, size, summary, description, build information and so forth.
-l lists all files in the package.
-s displays the state of each file in the package. The state is either normal, not installed or replaced.
-d displays all files that are listed as documentation.
-c displays all files that are listed as configuration files.

With these options you can do a number of great things. Below are some examples:
• Do you want to know which package the dig program is in? Try rpm -qf `which dig` or rpm -qif `which dig`
• Need to know what documentation is available for a specific command, and man -k commandname does not work? Try rpm -qdf `which nslookup`
• Need a lot of data to test a network connection? Try rpm -qila
• Need to know which not yet installed RPM package file contains the program "pico"?
Sorry, you are out of luck here. RPM only queries one rpm package at a time, so you need to do something like this:

```
for package in *.rpm
do
rpm -q -l -p $package | grep -q pico
if [ $? = 0 ]
then
echo $package
fi
done
```

rpmdb Database (Red Hat only)

- rpmdb-version.rpm: Database of all capabilities that all RPMs provide
- Allows you to use the **--redhatprovides** option

```
# rpm -iv rpmdb-redhat-7.0-0.20000830.i386.rpm
rpmdb-redhat

# rpm -iv xboard-4.0.7-3.i386.rpm
error: failed dependencies:
 chessprogram is needed by xboard-4.0.7-3
# rpm -q --redhatprovides chessprogram
gnuchess-4.0.pl80-6
# rpm -iv gnuchess-4.0.pl80-6.i386.rpm
gnuchess
# rpm -iv xboard-4.0.7-3.i386.rpm
xboard
```

Note: SuSE solves dependency conflicts through YaST

Notes:

The dependency information that is used by the RPM system is not based on actual package names, but rather on capabilities. This is done because multiple packages might actually offer the same capability. Suppose for instance that a certain package requires the availability of a mail reader. Then it doesn't matter whether pine, elm, mail or mailx is installed, as long as at least one of these is present. This works fine, but obviously makes it a little difficult to determine which packet to install if a certain capability is missing.

Red Hat has solved this with the rpmdb database. What basically happens is that, when the distribution is created, all rpm files are queried for the capabilities they provide. This is stored in the rpmdb database, which is an rpm file itself and can be installed like any other rpm. When installed, this database can be queried using the --redhatprovides option.
Starting with 8.0, Red Hat automatically suggests packages if a capability is missing, but the rpmdb database is installed and the capability is listed there. Unfortunately, there is no way of automatically installing all required packages.
SuSE does not solve this within the rpm program, but instead has integrated automatic dependency checking in yast. yast also installs all the missing RPMs automatically.

RPM Verifying

- Verifies the actual files with the original RPM
  - size S
  - MD5 checksum 5
  - permissions,type M
  - owner U
  - group G
  - modification time T
  - symbolic link L
  - device D

```
# rpm -V kdelibs3
.M......  /opt/kde3/kpac_dhcp_helper
.......T  /opt/kde3/share/mimelnk/application/x-applix.desktop
```

a dot (.) means: test passed

Notes:

The verify option verifies all files that are supposed to be present in the RPM against the files that are available on disk. This is a very easy way to check for any unauthorized configuration changes.

The following checks are performed on each file in an RPM:
5 MD5 checksum. This is a very hard to fool checksum which checks whether the contents of a file have changed.
S File size. This checks whether the size of the file has changed.
L Symbolic link. This verifies whether a certain symlink has changed.
t File modification time. This checks whether the file modification timestamp (mtime) has changed.
d Device. This verifies whether the major and minor numbers of a device are still intact.
U User. Is the owner of the file still the same?
G Group. Is the group of the file still the same?
M Mode. Are permissions, SUID, SGID bits and the file type still the same?

If a file checks out ok, there will be no output. If there is a discrepancy however, the name of the involved file will be listed, prepended by the discrepancy information. The output line will then look like this:
rpm -V sendmail
SM5....T c /etc/sendmail.cf

This means that a discrepancy was found in the file /etc/sendmail.cf. This is to be expected, since this file is a configuration file (hence the "c" in the line. The discrepancy information in this case is SM5....T, in which each letter denotes a certain discrepancy from the list above. In this case the following discrepancies were found: size, mode, MD5 checksum, modification time.

RPM Signatures

- RPM's can be signed by the distributor
- To verify signature:
  - Obtain public key of distributor
    - CD-ROM
    - Internet
  - Add public key to keyring using **gpg --import** (RPM v3) or **rpm --import** (RPM v4)
  - Verify package with **rpm --checksig**

```
redhat# rpm --import /mnt/cdrom/RPM-GPG-KEY
suse# gpg --import /mnt/cdrom/pubring.gpg

# rpm --checksig passwd-0.64.1-1.i386.rpm
passwd-0.64.1-1.i386 md5 gpg OK
```

Note: You can list the installed keys with "gpg --list-keys" (RPM v3) or "rpm -qa gpg-pubkey*" (RPM v4)

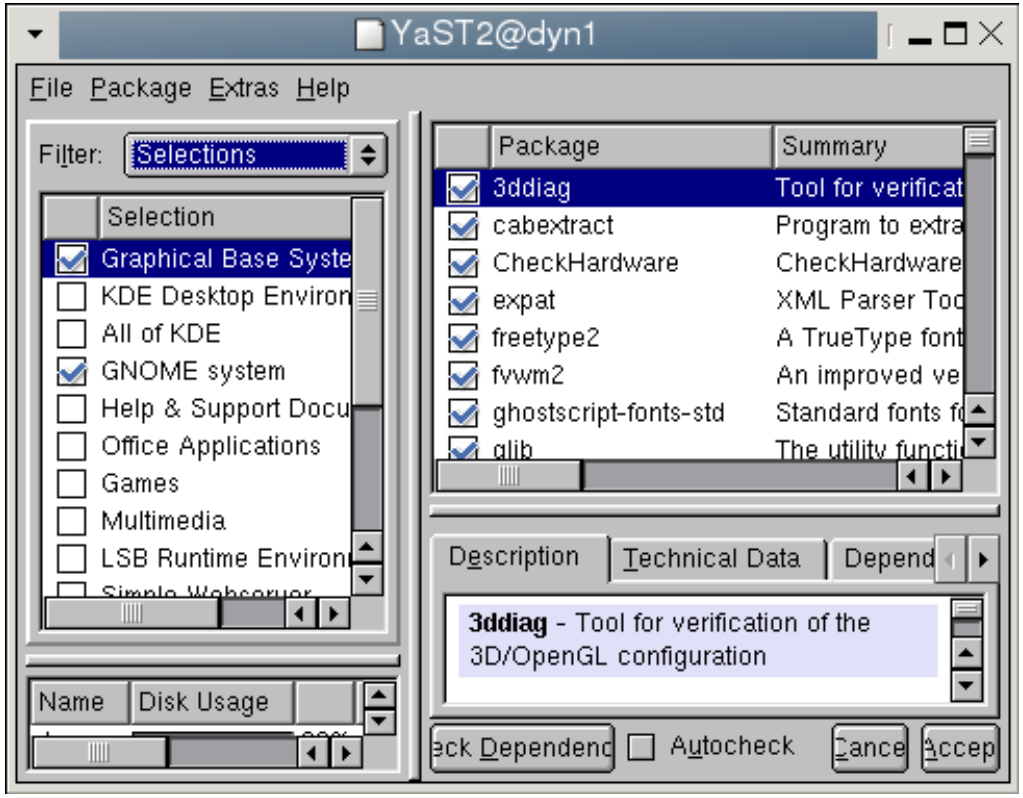
Notes:

The RPM Package format also features the ability to include a digital signature of a package, and most distribution builders actually make use of this feature as an effective measure against trojan horses introduced in an RPM after release by the distribution builder.

Verifying this signature is a two-step process. The first step is to obtain the public key of the distribution builder. This key is stored in a text file which can usually be found on the original CD-ROMs or on the distribution website. This public key needs to be added to your "keyring", your database of public and secret keys in your home directory. This is done with the following command: rpm --import /mnt/cdrom/RPM-GPG-KEY. Note that some distributions (for instance, SuSE), perform this step automatically while installing.

The second step is to verify each individual package. This is done with the command rpm --checksig packagename. If the output is "gpg OK", then you can be sure that it was indeed the distribution builder that built this individual package, and that no one has tampered with it since.

Integrated Package Management



redhat-config-packages

yast (install and remove software)

Notes:

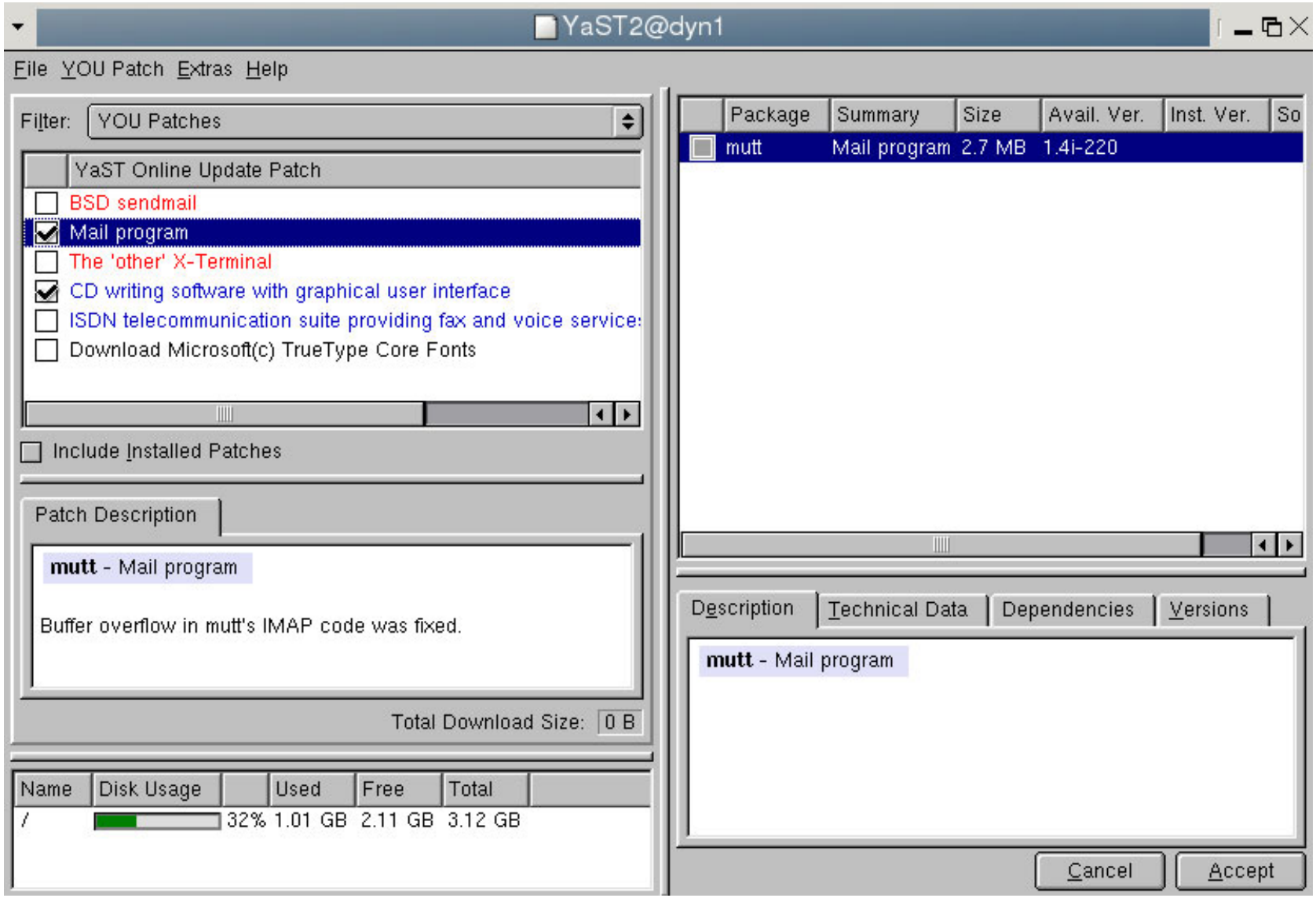
Keeping Up To Date (Red Hat)

- "Red Hat Network" (RHN)
  - Free and commercial subscriptions available
  - Create and manage account and systems on http://rhn.redhat.com
  - Register individual systems with **up2date --register**
  - Use **up2date** to bring system up to date, or use web interface at http://rhn.redhat.com (requires **rhnsd** daemon running on system)

Notes:

Keeping Up To Date (SuSE)

- **you** (Yast Online Update): Program that downloads/installs patches from any SuSE mirror



Notes:

Debian Package Management

- Debian uses its own package format and management tools

	Red Hat	Debian
Filenames	hello-1.0-1.src.rpm hello-1.0-1.i386.rpm	hello_1.0-1.tar.gz hello_1.0-1.deb
Install Upgrade Deinstall Query	rpm -i rpm -U, rpm -F rpm -e rpm -q	dpkg -i dpkg -i dpkg -r dpkg -p
Front-ends	redhat-config-packages yast, up2date, you	apt-get dselect
Package descriptor file	hello.spec	hello/debian/control hello/debian/rules
Build a package	{rpm|rpmbuild} -b	dpkg -b, dpkg-buildpackage

- To convert between DEB and RPM packages use Alien

Notes:

The Red Hat Package Manager

Section O12

Lab

Exercise Instructions With Hints

Retrieving information about installed packages

__ 1. Make a list of all packages that are installed on the system.
» **# rpm -q -a**

__ 2. Find out which package installed the /etc/inittab file.
» **# rpm -q -f /etc/inittab**

__ 3. List the information of that package.
» **# rpm -q -i -f /etc/inittab**

__ 4. List all files in that package.
» **# rpm -q -l -f /etc/inittab**

__ 5. Verify whether all files in that package are still the same. Which file has changed and in what respect? Why?
» **# rpm -V -f /etc/inittab**

Installing packages

__ 6. Create a directory /mnt/install. Mount the installation directory from the installation server on /mnt/install. List all the package files that are available on the install server.
» **# mkdir /mnt/install**
» **# mount server: directory /mnt/install**
» **# cd /mnt/install**
» **# find . -name "*.rpm" -print**

__ 7. Add the public key from the distribution to your keyring.
» redhat**# rpm --import RPM-GPG-KEY**
suse**# gpg --import pubring.gpg**

__ 8. Verify that the package xsnow is not installed. Verify the package on the CD, and install it. Then verify that it installed, and list the files in the package.
Note: Distributions change, and xsnow might not be included on your CD. In that case, use another entertaining X application, such as xearth, xjewel, xhangman or xbill.
» **# xsnow**
» **# rpm -q xsnow**
» **# rpm -K xsnow- version.rpm**
» **# rpm -ivh xsnow- version.rpm**
» **# rpm -qil xsnow**
» **# xsnow**

__ 9. Deinstall the xsnow package
» **# rpm -e xsnow**

Using a package management frontend

__ 10. If necessary, start X. Then, start the preferred RPM frontend tool for your distribution. Use this to install xsnow again.
Note: Red Hat does not include **GnoRPM** or **kpackage**, and its own frontend tool (**redhat-config-packages**) can only install packages from package groups that are predefined in RedHat/base/comps.xml on CD1. xsnow is not in any of these groups,so there is no frontend tool on Red Hat that allows you to install xsnow.
On Red Hat, use **redhat-config-packages** to install a package that is in one of the predefined groups.
» redhat**# redhat-config-packages -t /mnt/install**
suse**# yast2**