

Chapter 6 RPM - Notes

6.2 Introduction

Red Hat Package Manager (RPM) used by number of major distributions (and close relatives) to control installation, verification, upgrade, removal of software on Linux systems. Low-level **rpm** program performs all these operations, either on just one package or on list of packages. Operations that would cause problems (eg. removing package that another package depends on, or installing package when system needs other software to be installed first) blocked from completion.

6.3 Learning Objectives:

- Understand how the **RPM** system is organized and what major operations the **rpm** program can accomplish.
- Explain the naming conventions used for both binary and source **rpm** files.
- Query, verify, install, uninstall, upgrade, and freshen packages.
- Grasp why new kernels should be installed, rather than upgraded.
- Use **rpm2cpio** to copy packaged files into a **cpio** archive, as well as to extract the files without installing them.

6.4 RPM

RPM (Red Hat Package Manager) developed by Red Hat (unsurprisingly). All files related to specific task packaged into single **rpm** file, which also contains information about how/where to install/uninstall files. New versions of software lead to new **rpm** files, which then used for updating.

rpm files also contain dependency information. Note: unless given specific **URL** to draw from, **rpm** does not retrieve packages over network, installs only from local machine using absolute/relative paths.

rpm files usually distribution-dependent; installing package on different distribution than it was created for -> difficult, if not impossible.

6.5 Advantages of Using RPM

For system administrators, RPM makes it easy to:

- Determine what package (if any) any file on system is part of
- Determine what version installed
- Install/uninstall (erase) packages without leaving debris behind
- Verify that package was installed correctly. Useful for both troubleshooting/system auditing
- Distinguish documentation files from rest of package, optionally decide not to install them to save space
- Use **ftp** or **HTTP** to install packages over internet

For developers, RPM also offers advantages:

- Software often made available on more than one operating system. With RPM, original full + unmodified source used as basis, but developer can separate out changes needed to build on Linux
- More than one architecture can be built using only one **source package**

6.6 Package File Names

RPM package file names based on fields representing specific information, as documented in [RPM standard](#):

- Standard naming format for binary package:

```
<name>-<version>-<release>.<distro>.<architecture>.rpm
```

```
sed-4.2.2-5.el7.x86_64.rpm
```

- Standard naming format for source package:

```
<name>-<version>-<release>.<distro>.src.rpm
```

```
sed-4.2.2-5.el7.src.rpm
```

Note: **distro** field often actually specifies repository that package came from, as given installation may use number of different package repositories, as discussed for **yum/zypper** which work above RPM.

6.7 Database Directory

`/var/lib/rpm` -> default system directory that holds RPM database files in form of Berkeley DB hash files. Database files should not be manually modified, updated should be done only through use of **rpm** program.

Alternative database directory can be specified with `--dbpath` option to **rpm** program. Might do this, for example, to examine RPM database copied from another system.

Can use `--rebuilddb` option to rebuild database indices from installed package headers. More of repair, not rebuild from scratch.

6.8 Helper Programs and Modifying Settings

Helper programs and scripts used by RPM reside in `/usr/lib/rpm`. Quite a few, eg. on RHEL 7 system:

```
$ ls /usr/lib/rpm | wc -l
73
```

where **wc** reporting number of lines of output.

Can create `rpmrc` file to specify default settings for **rpm**. By default, **rpm** looks for:

1. `/usr/lib/rpm/rpmrc`
2. `/etc/rpmrc`
3. `~/.rpmrc`

in above order. Note: all these files read; **rpm** does not stop as soon as it finds that one exists. Alternative `rpmrc` file can be specified using `--rcfile` option.

6.9 Queries

All **rpm** inquiries include `-q` option, which can be combined with numerous sub-options:

- Which version of a package is installed?

```
$ rpm -q bash
```

- Which package did this file come from?

```
$ rpm -qf /bin/bash
```

- What files were installed by this package?

```
$ rpm -ql bash
```

- Show information about this package.

```
$ rpm -qi bash
```

- Show information about this package from the package file, not the package database.

```
$ rpm -qip foo-1.0.0-1.noarch.rpm
```

- List all installed packages on this system.

```
$ rpm -qa
```

Couple of other useful options: `--require` , `--whatprovides` :

- Return a list of prerequisites for a package:

```
$ rpm -qp --requires foo-1.0.0-1.noarch.rpm
```

- Show what installed package provides a particular requisite package:

```
$ rpm -q --whatprovides libc.so.6
```

6.10 Verifying Packages

`-v` option to **rpm** allows you to verify whether files from particular package consistent with system's RPM database. To verify all packages installed on system:

```
$ rpm -Va
missing /var/run/pluto
....
S.5....T. c /etc/hba.conf
S.5....T. /usr/share/applications/defaults.list
....L.... c /etc/pam.d/fingerprint-auth
....L.... c /etc/pam.d/password-auth
....
.M..... /var/lib/nfs/rpv-pipefs
....
....UG.. /usr/local/bin
....UG.. /usr/local/etc
```

show ing just few items. (Note: command can take long time, as it examines all files ow ned by all packages.)

Output generated only w hen there is problem.

Each characters displayed denotes result of comparison of attribute(s) of file to value of those attribute(s) recorded in database. Single . (period) means test passed, w hile single ? (question mark) indicates test could not be performed (eg. file permissions prevent reading). Otherw ise, character denotes failure of corresponding verification test:

- **S**: file size differs
- **M**: file permissions and/or type differs
- **5**: MD5 checksum differs
- **D**: device major/minor number mismatch
- **L**: symbolic link path mismatch
- **U**: user ow nership differs
- **G**: group ow nership differs
- **T**: modification time differs
- **P**: capabilities differ

Note: many verification tests do not indicate problem. Eg. many configuration files modified as system evolves.

If specifying one or more package names as argument, examine only that package:

- No output w hen everything is OK:

```
$ rpm -V bash
```

- Output indicating that a file's size, checksum, and modification time have changed:

```
$ rpm -V talk
S.5....T in.ntalkd.8
```

- Output indicating that a file is missing:

```
$ rpm -V talk
missing /usr/bin/talk
```

6.11 Installing Packages

Installing package as simple as:

```
$ sudo rpm -ivh foo-1.0.0-1.noarch.rpm
```

where **-i** for install, **-v** for verbose, **-h** to print hash marks to show progress.

RPM performs number of tasks w hen installing package:

- Performs dependency checks: necessary because some packages w ill not operate properly unless one or more other packages also installed.
- Performs conflict checks: include attempts to install already-installed package or to install older version over new er version.

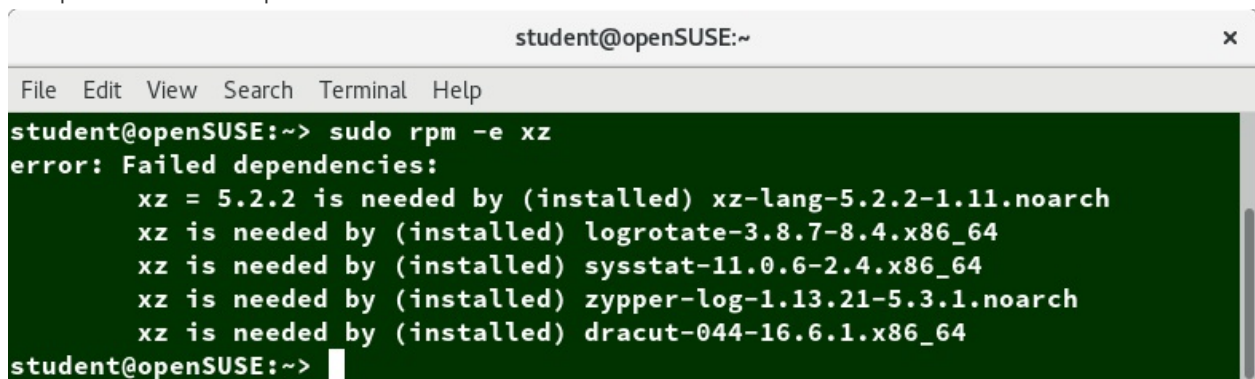
- Executes commands required before installation: developer building package can specify that certain tasks be performed before or after install.
- Deals intelligently with configuration files: when installing configuration file, if file exists and has been changed since previous version of package was installed, RPM saves old version with suffix `.rpmsave`. Allows you to integrate changes made to the configuration file into new version of file. Feature depends on properly created RPM packages.
- Unpacks files from packages, installs them with correct attributes: in addition to installing files in right place, RPM also sets attributes such as permissions, ownership, modification (build) time.
- Executes commands required after installation: performs any post-install tasks required for setup or initialization.
- Updates system RPM database: every time RPM install package, updates information in system database. Uses information when checking for conflicts.

6.12 Uninstalling Packages

`-e` option causes **rpm** to uninstall (erase) package. Normally, `rpm -e` fails with error message if package attempting to install either not actually installed, or required by other packages on system. Successful uninstall produces no output.

```
$ sudo rpm -e system-config-lvm
package system-config-lvm is not installed
```

Example of error due to dependencies:



```
student@openSUSE:~
File Edit View Search Terminal Help
student@openSUSE:~> sudo rpm -e xz
error: Failed dependencies:
    xz = 5.2.2 is needed by (installed) xz-lang-5.2.2-1.11.noarch
    xz is needed by (installed) logrotate-3.8.7-8.4.x86_64
    xz is needed by (installed) sysstat-11.0.6-2.4.x86_64
    xz is needed by (installed) zypper-log-1.13.21-5.3.1.noarch
    xz is needed by (installed) dracut-044-16.6.1.x86_64
student@openSUSE:~>
```

Can use `--test` option alone with `-e` to determine whether uninstall would succeed or fail, without actually doing uninstall. If operation successful, **rpm** prints no output. Add `-vv` option to get more information.

Remember: package argument for erase is package name, not **rpm** file name.

Important (but obvious) note: Never remove (erase/uninstall) **rpm** package itself. Only way to fix this problem to re-install operating system, or booting into rescue environment.

6.14 Upgrading Packages

Upgrading replaces original package (if installed):

```
$ sudo rpm -Uvh bash-4.2.46-30.el7_0.4.x86_64.rpm
```

Can give list of package names, not just one.

When upgrading, already installed package removed after newer version installed. One exception: configuration files from original installation -> kept with `.rpmsave` extension.

If `-u` option used and package not already installed, simply installed and no error.

`-i` option not designed for upgrades. Attempting to install new RPM package over older one fails with error messages, because it tries to overwrite existing system files.

Different versions of same package may be installed if each version of package does not contain same files: kernel packages and library packages from alternative architectures typically only packages that would be commonly installed multiple times.

If want to downgrade with `rpm -U` (to replace current version with earlier version), must add `--oldpackage` option to command line.

6.15 Freshening Packages

```
$ sudo rpm -Fvh *.rpm
```

will attempt to **freshen** all packages in current directory:

1. If older version of package installed, will be upgraded to newer version in directory
2. If version on system is same as one in directory, nothing happens
3. If no version of package installed, package in directory ignored

Freshening useful for applying lot of patches (ie, upgraded packages) at once.

6.16 Upgrading the Kernel

When installing new kernel on system, requires reboot (one of few updates that do) to take effect. Should not do upgrade (`-u`) of kernel: upgrade would remove old currently running kernel.

This in and of itself won't stop system, but if, after reboot, have any problems, will no longer be able to reboot into old kernel, since removed from system. However, if install (`-i`), both kernels coexist and can choose to boot into either one, ie. can revert back to old one if need be.

To install new kernel:

```
$ sudo rpm -ivh kernel-{version}.{arch}.rpm
```

filling in correct version + architecture names.

When doing this, GRUB configuration file automatically updated to include new version. Will be default choice at boot, unless reconfigure system to do something else.

One new kernel version tested, may remove old version if you wish, though not necessary. Unless short on space, recommended to keep one or more older kernels available.

6.17 Using rpm2cpio

Suppose have need to extract files from **rpm** but do not want to actually install package?

rpm2cpio program used to copy files from **rpm** to **cpio** archive + extract files if desired.

Create **cpio** archive with:

```
$ rpm2cpio foobar.rpm > foobar.cpio
```

To list files in **rpm**:

```
$ rpm2cpio foobar.rpm | cpio -t
```

but better way is to:

```
$ rpm -qilp foobar.rpm
```

To extract on system:

```
$ rpm2cpio bash-XXXX.rpm | cpio -ivd bin/bash  
$ rpm2cpio foobar.rpm | cpio --extract --make-directories
```

##

[Back to top](#)

[Previous Chapter](#) - [Table of Contents](#) - [Next Chapter](#)