Chapter 22: RPM Package Management

Introduction to System Software

- Linux distributions vary considerably in the which package management system is used for managing system software
- The two most common package management systems were started by two of the most popular distributions:
  - RPM Package Management is from Red Hat
  - Debian Package Management is from Debian

Package Management Systems

- Provide ability to download, install, configure, query, update and remove software
- Ensures requirements of software called *dependencies* are maintained
- Each system has its own advantages and disadvantages, but all systems function in a very similar way

RPM Package Management

- Originally Red Hat Package Management
- Two types of files, source and binary:
  - ".src.rpm" used to compile on any architecture
  - ".rpm" pre-compiled for a specific architecture
- Compiled ".rpm" Naming convention:
  - package-name-version-release.architecture.rpm
  - Example: x3270-x11-3.3.6-10.5.el6.i686.rpm

RPM

- The basic level command for managing RPM files is the `rpm` command
- The `rpm` command can be used even when networking is unavailable
- Resolving dependencies between software packages is more difficult with the `rpm` command than it is with the `yum` command
- The `rpm` command is able to perform many queries related to software packages

rpm Queries

- The `rpm` command can query information about:
  - packages installed on the system
  - RPM files that are either on the filesystem or reachable by an URL
- Querying a package file that is not installed:
  - requires adding the `-p` option to the query
  - requires using the full pathname of the file instead of just the package name

Querying Information on a RPM

- To display information on an installed rpm:
  #rpm -qi bash

```
$ rpm -qi bash
Name        : bash                         Relocations: (not relocatable)
Version     : 4.1.2                           Vendor: CentOS
Release     : 15.el6_4                        Build Date: Thu 18 Jul 2013 02:21:24
 PM BST
Install Date: Mon 09 Jun 2014 04:12:34 PM BST      Build Host: c6b10.bsys.dev.ce
ntos.org
Group       : System Environment/Shells     Source RPM: bash-4.1.2-15.el6_4.src.
rpm
Size        : 3139291                          License: GPLv3+
Signature   : RSA/SHA1, Thu 18 Jul 2013 02:46:10 PM BST, Key ID 0946fca2c105b9de
Packager    : CentOS BuildSystem <http://bugs.centos.org>
URL         : http://www.gnu.org/software/bash
Summary     : The GNU Bourne Again shell
Description : The GNU Bourne Again shell (Bash) is a shell or command language interpreter
that is compatible with the Bourne shell (sh). Bash incorporates useful features from the Korn
shell (ksh) and the C shell (csh). Most sh scripts can be run by bash without modification.
```

Querying Information on a RPM

- To display information about an RPM file:
  #rpm -qip bash-4.1.2-14.el6.i686.rpm

Querying an Installed RPM

- To query whether a package is installed, a user can execute:

  #rpm -q *package-name*
- If the package is installed, the output will be: *package-name*-version-release.architecture
- If the package is not installed, the output will be: package *package-name* is not installed

Querying all installed RPMs

- To display a list of all installed RPMs:
  #rpm -qa
  - sorted alphabetically: **#**rpm -qa | sort
  - sorted chronologically: **#**rpm -qa --last
- To display a list filtered by a glob:
  #rpm -qa "*x11*"
  - shows any package containing "x11" in its name

RPM Queries

Option	Purpose
-a	Used to list all installed packages
-c	Display list of configuration files that belong to the package
-d	List the documentation files that belong to the package
-i	Display the package information
-K	Check the package integrity
-l	List all files in the package
--provides	List capabilities on which this package depends
-R	List capabilities that this package requires
--scripts	List the scripts that are used before and after installation of the package
-s	Display state of each package file as normal, not installed or replaced

RPM Scripts

- RPM files may contain commands to execute during installation or removal
- As installation or removal of an RPM normally requires root privileges, these scripted commands will be executed as the root user which can be dangerous.
- To inspect the scripts:

  #rpm -qp --scripts package-version-release-arch.rpm

RPM Integrity

- RPM files are *signed* by the private key of the distributor, who publishes the corresponding public key
- After importing the public key, `rpm` can verify the integrity of an RPM signed by the matching private key
- Red Hat-derived distributions store public keys in the /etc/pki/rpm-gpg directory

Verifying RPM Integrity

- Import the public keys:
  #rpm --import /etc/pki/rpm-gpg/*
  #rpm --import URL-address-of-public-key
- Verify the package is OK:
  - rpm -qpK package-version-release-arch.rpm
  - Valid RPM output: sha1 (md5) pgp md5 **OK**
  - Corrupt RPM: rsa sha1 **(MD5) PGP MD5 NOT OK**

Installing packages with rpm

- Unlike queries, installation of an RPM normally requires root privileges
- Using the `rpm` command to install an RPM package will fail if the dependencies for that package are not installed
- The error message when installation fails can be helpful to locate the missing dependencies

rpm Installation Example

- #rpm -i gnome-games-2.28.2-2.el6.i686.rpm

```
error: Failed dependencies:
    libclutter-glx-1.0.so.0 is needed by gnome-games-
1:2.28.2-2.el6.i686
    libguile.so.17 is needed by gnome-games-1:2.28.2-
2.el6.i686
```

- Search for these file names suggests installing the “clutter” and “guile” packages
- #rpm -i gnome-games-2.28.2-2.el6.i686.rpm clutter-1.0.6-3.el6.i686.rpm guile-1.8.7-5.el6.i686.rpm

Removing Packages With rpm

- Remove packages with the -e option for the `rpm` command
- You can not remove a package that another package depends on, attempting to do so will result in an error:

```
error: Failed dependencies:
    x3270 = 3.3.6 is needed by (installed) x3270-x11-
3.3.6-10.5.el6.i686
    libicuuc.so.42 is needed by (installed) x3270-x11-

3.3.6-10.5.el6.i686
```

Updating Packages with rpm

- Use -U to either update to a new version or to install
- Use -F to only update
- Kernel packages should never be updated; always use -i to install

Using rpm2cpio

- Use `rpm2cpio` to extract individual files from a package or list contents
- Takes package as input and produces a cpio format as output
- Display list of package contents example:

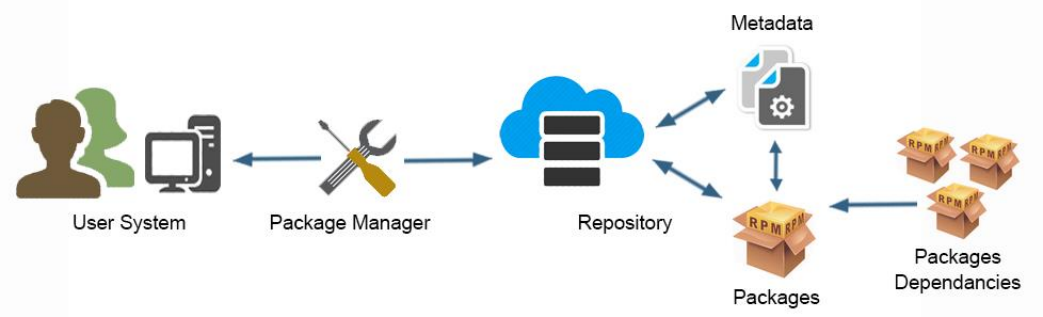
```
#rpm2cpio telnet-server-0.17-47.el6_3.1.i686.rpm
  | cpio -t
```

- Extract contents of package example:

```
#rpm2cpio telnet-server-0.17-47.el6_3.1.i686.rpm
  | cpio -iumd
```

Manage packages with yum

- Designed as a "Front end" for rpm
- yum advantages:
  - can automatically resolve dependencies
  - can download packages from a repository server
  - can query package information from repository server



yum Basics

- Configuration:
  - the /etc/yum.conf file
  - the /etc/yum.repos.d directory
- The `yumdownloader` command allows you to download packages from the repositories that are configured

Determining Dependencies

- Finding dependency packages is difficult with `rpm`
- Use the **provides** subcommand to `yum`:

```
#yum provides /usr/lib/libicuuc.so.42
```

- Search for key terms by using the **search** subcommand to `yum`:

```
#yum search terminal
```

Installing With yum

- To install package:
  $ `yum install package`
- To install package without prompts:
  $ `yum -y install package`
- To install groups:
  $ `yum groupinstall group_name`
- To list group names:
  $ `yum grouplist`
- To list group information:
  $ `yum groupinfo group_name`

Removing With yum

- To install a package:
  - **$**`yum remove package`
  - **$**`yum erase package`
  - Both perform same operation
- Use -y to remove without prompts (not recommended)
- Removes all dependency packages as well

Update With yum

- To list packages that need to be updated:

```
$yum list updates
```

- To update package:

```
$yum update package
```

dnf

- Dandified YUM
- Since 2013
- Commands are backward compatible with yum

Introduction to Debian Package Management

- Used on several distributions, including Ubuntu and Mint
- The ".deb" files are binary files compiled to execute on a particular computer architecture
- The source packages, which contain the original source code, have an extension of ".dsc"

Managing Repositories

- Package files are also commonly installed by downloading them directly from repositories located on Internet servers
- The primary configuration file that is used to locate these repositories is the /etc/apt/sources.list file
- Run `apt-get update` if source.list file is modified

Debian Package Tools

- Basic command: `dpkg` (akin to rpm)
- Front end commands:
  - `synaptic` (command line)
  - `aptitude` (provides GUI interface)
  - Advanced Package Tool (APT)

Installing Software with dpkg

- Exmple:

  $dpkg -i firefox_36.0.1+build2-0ubuntu0.14.04.1_amd64.deb
- The `dpkg` command does not provide any automatic way to resolve dependency issues
- The APT tool `apt-get` does resolve dependencies

Removing Software with dpkg

- Two options to remove software:
  - The -r option removes most of the software but leaves the configuration files in case it is reinstalled
  - The -P (purge) option removes all of the software and configuration files

Listing Software with dpkg

- Use the -l option to list all currently installed packages

```
$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-
   pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name          Version      Architecture Description
+++-=============-============-============-=============
ii  adduser       3.113+nmu3ub all          add and remove users and groups
ii  apt           1.0.1ubuntu2 amd64        command-line package manager
ii  apt-file      2.5.2ubuntu1 all          search for files within Debian pa
ii  apt-utils     1.0.1ubuntu2 amd64        package management related utilit
ii  apt-xapian-ind 0.45ubuntu4  all         maintenance and search tools for
```

Listing Software with dpkg

- Output (previous slide) contains five pieces of information:
  - Desired status of the package ("i" for installed, "u" for unknown, "r" for remove and "h" for hold)
  - Actual status of the package ("i" means installed and"n" means not installed)
  - The package name, version (combined with the release) and the description of the package

Listing Software with dpkg

- Use the -L option to display files that are installed with software package
- Use the -S option with a file name as an argument to list what package provided the file

Reconfigure packages

- Packages are configured when they are installed
- To reconfigure a package:
  - dpkg-reconfigure *package*
  - Exmple: **$**dpkg-reconfigure tzdata

Searching for Packages With apt-cache

- Display packages that need to be updated:
  #apt-get *update*
- Search repository for packages:
  #apt-cache search *keyword*
- Display package dependencies:
  #apt-cache depends *package*
- Display package status:
  #apt-cache show *package*

Installing/Updating With apt-get

- apt-get automatically resolves dependencies:
  #apt-get update *package*
- Install or update: apt-get install *package*
- Only update: apt-get --only-upgrade install *package*
- Upgrade all packages that need updating:
  #apt-get update
  #apt-get upgrade

Removing With apt-get

- Remove, but keep configuration files in case reinstalled:

  #apt-get remove *package*
- Remove, including configuration files:

  #purge remove *package*