Linux for Researchers

Chapter 10: Installation and Package Management

In earlier years, I'd have begun this series by talking about installing Linux. Installing Linux then was a complicated, hit-or-miss process. Maybe it would work, maybe it wouldn't, but you'd spend a lot of time on it and you'd learn way more about computers than you really wanted to know in the process.

These days, installation has become something of a non-issue. You can buy computers from many vendors with Linux pre-installed. If you do need to install Linux yourself, all major Linux distributors provide easy-to use CD or DVD images that will do the installation for you quickly and painlessly.

Part 1: Distributions:

There are hundreds of Linux distributions available. Some are commercial, some are free, and some are a hybrid where, for example, you get the distribution for free, but can pay for support.

All the most popular Linux distributions today first appeared around 1993, or are direct descendants of such distributions. The Red Hat distribution (one of the most popular, especially in the corporate world) was created by Marc Ewing in 1994, while he was a student at CMU. Red Hat spawned Fedora, CentOS and others.

The Debian distribution was created by Ian Murdock in 1993. It is the foundation for the Ubuntu distribution, created by Mark Shuttleworth's company, Canonical Ltd.

What's a Distribution?:

A Linux “distribution” is a kit that can be used to:

- Partition and format a computer's hard disk,
- Install the Linux kernel and a boot loader,
- Install a suite of software.

Distributions differ in the selection of software they offer, and in the look and feel of the installation process. They also differ in the quality and quantity of post-installation support they offer.

Linux distributions tend to include much more software than Windows or OS X installation kits.

In addition to the basic tools like shells, calculators and web browsers, Linux distributions include word processors, spreadsheets, a large number of compilers and interpreters, database servers, web servers, and so on.

You can find a “family tree” of Linux distributions here:
https://commons.wikimedia.org/wiki/File:Linux_Distribution_Timeline.svg

Popular Distributions:

Among the most popular distributions are:

Red Hat Enterprise Linux (RHEL):
This is a commercial product, but the source code for it is free, and has been re-compiled and released through other distributions (such as CentOS) for free. RHEL aims to be an “enterprise” distribution: it changes slowly, and each release is supported for many years.

Fedora:
Fedora is a free distribution overseen by Red Hat. It is used as the basis for RHEL. Fedora changes rapidly, and each release is only supported for a few months.

Debian:
Debian is a free distribution maintained by highly-structured community of developers, woldwide. Administrators are selected through periodic elections, and decisions about the distribution are made by these elected officials and through General Resolutions voted on during the elections. The Debian distribution has a reputation for reliability and slow (often cited as “glacial”) change. It also offers more software than most distributions.

Ubuntu:
Ubuntu is a free distribution, with commercial support, overseen by Canonical, Ltd. Ubuntu is based on Debian, with many additions and changes. Through its participation in the Debian community, Canonical has sped up and regularized the Debian release cycle. New Ubuntu releases appear frequently, and occasionally a release is tagged for “Long Term Support”. These “LTS” releases are supported for several years.

Which distribution is best? It depends on your needs.

Choosing a Distribution:

Some factors to consider when deciding which distribution to use:

- Support lifetime:
If a distribution only offers security fixes for 18 months (like Fedora), then your computer will need to be upgraded every 18 months. For a single computer, this may be OK. If you need to support many computers, it may not be.

- Quality of support:
Are security fixes and other bug fixes released quickly? Are the distribution's maintainers responsive to bug reports?

- Support from 3rd-party vendors:
If you need to use software from 3rd-party vendors, do they provide software that will work with this distribution?

- Ease of maintenance and installation:
How easy is it to keep computers up-to-date with patches? How easy is it to deploy a new computer?

Comparison of Support Lifetimes:

These days, it's very important that security fixes be available as soon as possible after a new security flaw is found. Ideally, the maintainers of your Linux distribution should provide you with patches for new security flaws. Here's a table showing some of the most popular free Linux distributions, and the length of time during which security patches will be provided for a given version:

Distribution	Base	Support Life
CentOS	RHEL	10 yrs
ScientificLinux	RHEL	10 yrs
Ubuntu	Debian	10 yrs (LTS)
Fedora	self	1.5 yrs
SuSE	self	2 yrs
Debian	self	2 versions
Slackware	self	As possible

The “Base” column indicates which other distribution (if any) this distribution is based on.

You can see what the CentOS 7 installation process looks like here:

https://www.tecmint.com/centos-7-installation/

Later on, I'll point you to a similar demo showing an Ubuntu installation.

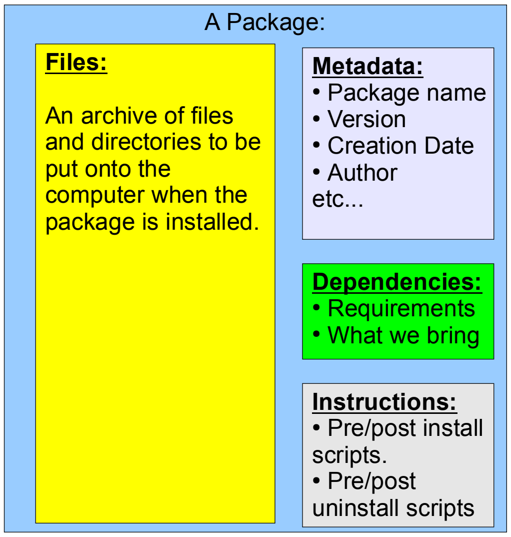
Part 2: Package Management Systems

Under most Linux distributions, software is bundled up into discreet “packages” that can be installed or removed separately. A package might contain a program and its associated documentation and configuration files, or it might contain some libraries or a bundle of fonts.

Several different package management systems are in common use among Linux distributions. A distribution's choice of package management system is very important. Packages need to be easy to install and delete, but they also need to be easy to create, to encourage vendors and developers to provide packages to you.

What's a Package?:

A package is an installation kit. The kit may have software bundled up inside it, or it may contain other types of files (or even just some instructions). A given package may require that other packages be installed first.


A Package:
Files:
An archive of files and directories to be put onto the computer when the package is installed.
Metadata:
• Package name
• Version
• Creation Date
• Author
etc...
Dependencies:
• Requirements
• What we bring
Instructions:
• Pre/post install scripts.
• Pre/post uninstall scripts

These “dependencies” are actually expressed in terms of “capabilities”. Each package provides certain capabilities, and requires certain capabilities to already be present.

A package may also contain metadata, such as a name for the package, a version number, etc.

Different Linux distributions use different package formats. Not all package formats have all of these features. The minimal format is just an archive of files and directories.

What's a Package Management System?:

Package management systems all perform at least some of these tasks:

- Allow you to install, uninstall or upgrade packages, warning about any unsatisified dependencies.

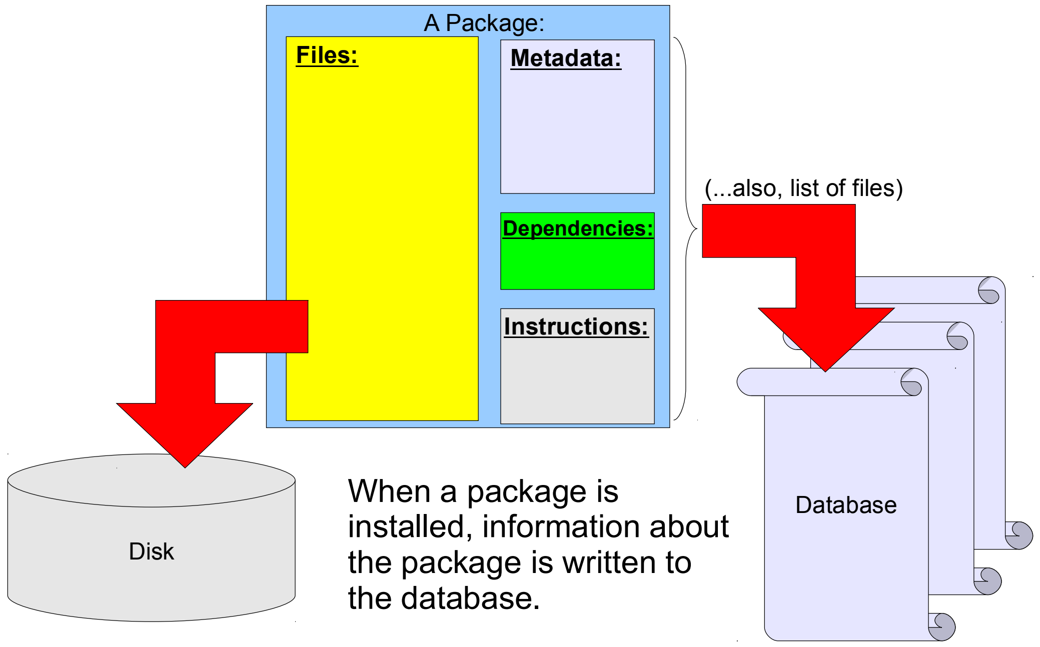
- Keep track of what packages are installed, and provide a list of their contents.

- Provide detailed information (e.g., version number, contents) about each installed package.

- Allow you to check to see if a package's files have been damaged since installation.

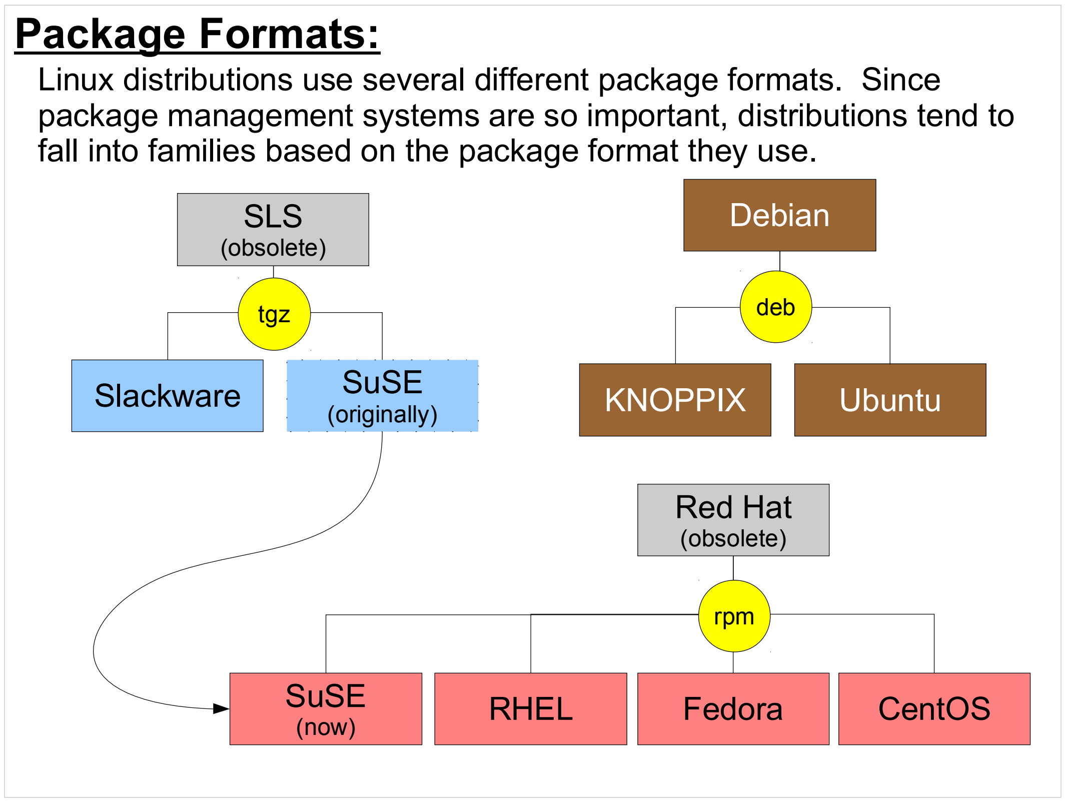
The Package Management Database:

To do all of these tasks, package managers typically maintain a database



When a package is installed, information about the package is written to the database.

For Red-Hat-based distributions, the database usually resides in /var/lib/rpm. For Debian-based distributions, you'll probably find it in /var/lib/dpkg.



Package Formats:

Linux distributions use several different package formats. Since package management systems are so important, distributions tend to fall into families based on the package format they use.

The “rpm” package format used in the Red Hat world and the “deb” package format used in the Debian world both include all the features we talked about earlier, including dependency information. The “tgz” format used in the Slackware world is much simpler, and contains no information about dependencies.

SuSE is the only major distribution that's changed its package format. It started out using the “tgz” format, then switched to “rpm”.

The original Red Hat distribution was called “Red Hat Linux” (RHL). This was discontinued in 2003. The successors were Fedora, which is completely free, and “Red Hat Enterprise Linux” (RHEL) which is a commercial product.

Part 3: The rpm Package Management System

Now let's take a detailed look at package management with one of these systems, rpm. At the end, I'll show you how specific rpm commands match up with analogous commands for the Debian (“deb”) package system. Rpm packages and deb packages are by far the most commonly used package formats for Linux.

Using the “rpm” Command:

Many Linux distributions use packages in the “Red Hat Package Manager” (rpm) format. You can use the “rpm” command to install, update, delete and query the status of these packages.

Installing a package:

```
[root@demo ~]# rpm -i firefox-3.0.6-1.el5.centos.i386.rpm
```

Querying a package, to see which version (if any) is installed:

```
[root@demo ~]# rpm -q firefox
firefox-3.0.6-1.el5.centos
```

Updating a package to a new version:

```
[root@demo ~]# rpm -U firefox-3.1.0-1.el5.centos.i386.rpm
```

Removing (“erasing”) a package:

```
[root@demo ~]# rpm -e firefox
```

Finding Package Information:

With the “-q -i” switches, you can get rpm to tell you a little more about an installed package. The firefox package we installed was a “binary RPM”. It contained software for us to use, but not the source code from which the software was compiled.

```
[root@demo ~]# rpm -q -i firefox
Name        : firefox
Relocations : (not relocatable)
Version     : 3.0.6
Vendor      : CentOS
Release     : 1.el5.centos
Build Date  : Wed 04 Feb 2009
Install Date: Thu 05 Feb 2009
Build Host  : sillage2.bis.pasteur.fr
Group       : Applications/Internet
Source RPM  : firefox-3.0.6-1.el5.centos.src.rpm
Size        : 15033628
License     : MPLv1.1 or GPLv2+ or LGPLv2+
Signature   : DSA/SHA1, Key ID a8a447dce8562897
Packager    : Tru Huynh <tru@centos.org>
URL         : http://www.mozilla.org/projects/fi
Summary     : Mozilla Firefox Web browser.
Description :
Mozilla Firefox is an open-source web browser,
designed for standards compliance, performance and
portability.
```

Packages are identified by a name, a version number and a release number. The version number is incremented by the software's author. The release number is incremented by the person who makes the package.

Binary packages are usually built from “source RPMs”. These bundle up the source code for a piece of software, along with instructions for building binary files from it.

RPM packages are signed, so you can verify that the package was created by someone you trust.

Getting Information About an RPM File:

You can use the “-p” switch to query an RPM file that hasn't been installed yet.

```
[root@demo ~]# rpm -q -i -p gimp-2.4.1-1.i386.rpm
Name        : gimp
Relocations : (not relocatable)
Version     : 2.4.1
Vendor      : (none)
Release     : 1
Build Date  : Fri 09 Nov 2007 11:03:43 AM EST
Install Date: (not installed)
Build Host  : galileo2.phys.virginia.edu
Group       : Applications/Multimedia
Source RPM  : gimp-2.4.1-1.src.rpm
Size        : 38273569
License     : GPLv2+
Signature   : (none)
URL         : http://www.gimp.org/
Summary     : GNU Image Manipulation Program
Description :
GIMP (GNU Image Manipulation Program) is a powerful image composition
and editing program, which can be extremely useful for creating logos
and other graphics for webpages. GIMP has many of the tools and filters
you would expect to find in similar commercial offerings, and some
interesting extras as well. GIMP provides a large image manipulation
toolbox, including channel operations and layers, effects, sub-pixel
imaging and anti-aliasing, and conversions, all with multi-level undo.
```

You can use the “-p” switch with most of these rpm commands to tell rpm to look at a file, rather than an already-installed package.

Listing All Installed Packages:

You can get a list of all installed packages by typing "rpm -q -a". It's often useful to pipe the output of this command into "grep" to look for a particular string. A typical Linux computer has thousands of packages installed.

```
[root@demo ~]# rpm -q -a
mktemp-1.5-23.2.2
perl-Compress-Zlib-1.42-1.fc6
perl-Digest-HMAC-1.01-15
libdaemon-0.10-5.el5
rootfiles-8.1-1.1.1
gail-1.9.2-1.fc6
libnotify-0.4.2-6.el5
opensp-1.5.2-4
at-3.1.8-82.fc6
im-chooser-0.3.3-6.el5
kdemultimedia-3.5.4-2.fc6
xorg-x11-drv-summa-1.1.0-1.1
gnuplot-4.0.0-12
jakarta-oro-2.0.8-3jpp.1
pango-devel-1.14.9-3.el5.centos
blas-3.0-37.el5
castor-test-0.9.5-1jpp.7
pyspi-0.6.1-1.el5
fonts-KOI8-R-75dpi-1.0-9.1.1
...etc.
```

You can use the "`--qf`" qualifier to control the output format:

```
[root@demo ~]# rpm -q -a \
--qf '%{NAME}: %{SUMMARY}\n'
mktemp: A small utility for safely
making /tmp files.
perl-Compress-Zlib: A module
providing Perl interfaces to the
zlib compression li
brary.
perl-Digest-HMAC: Digest-HMAC Perl
module
libdaemon: library for writing UNIX
daemons
rootfiles: The basic required files
for the root user's directory.
...etc.
```

You can get a list of available tags like "NAME" and "SUMMARY" by typing "`rpm --querytags`".

Which Package Installed a Given File?:

You can use the “-f” switch to find out which package a given file belongs to:

```
[root@demo ~]# rpm -q -f /bin/sh
bash-3.2-21.el5
```

You can use “-f” in combination with other switches to query package information without even knowing the package name:

```
[root@demo ~]#  rpm -q -f /bin/sh -i
Name        : bash
Relocations : (not relocatable)
Version     : 3.2
Vendor      : CentOS
Release     : 21.el5
Build Date  : Sat 24 May 2008 05:07:47 PM EDT
Install Date: Wed 25 Jun 2008 01:25:16 PM EDT
Build Host  : builder16.centos.org
Group       : System Environment/Shells
Source RPM  : bash-3.2-21.el5.src.rpm
Size        : 5349369
License: GPLv2+
...etc.
```

Listing the Files Installed by a Package:

You can use the “-l” switch to list the files installed by a given package:

```
[root@demo ~]# rpm -q -l firefox
/usr/bin/firefox
/usr/lib/firefox-3.0.6
/usr/lib/firefox-3.0.6/.autoreg
/usr/lib/firefox-3.0.6/LICENSE
/usr/lib/firefox-3.0.6/README.txt
/usr/lib/firefox-3.0.6/application.ini
/usr/lib/firefox-3.0.6/blocklist.xml
/usr/lib/firefox-3.0.6/browserconfig.properties
/usr/lib/firefox-3.0.6/chrome
/usr/lib/firefox-3.0.6/chrome/browser.jar
/usr/lib/firefox-3.0.6/chrome/browser.manifest
/usr/lib/firefox-3.0.6/chrome/classic.jar
/usr/lib/firefox-3.0.6/chrome/classic.manifest
/usr/lib/firefox-3.0.6/chrome/classic.jar
/usr/lib/firefox-3.0.6/chrome/classic.manifest
/usr/lib/firefox-3.0.6/chrome/en-US.jar
/usr/lib/firefox-3.0.6/chrome/en-US.manifest
/usr/lib/firefox-3.0.6/chrome/icons
...etc.
```

Listing Dependencies:

You can use the "--requires" or "--whatrequires" switches to find the prerequisites for installing a given package, or the list of packages that require the given package.

```
[root@demo ~]# rpm --requires gimp
/bin/bash ◄──────────────────── Files
rtld(GNU_HASH)◄──────────────── Arbitrary
                                Capabilities
desktop-file-utils >= 0.9 ⎫
fontconfig >= 2.2.0       ⎪
freetype >= 2.1.7         ⎬ Other Packages
glib2 >= 2.12.3           ⎪
gtk2 >= 2.10.13           ⎭
...etc.
```

This is the list of "capabilities" that need to be provided by other, already-installed packages before this package can be installed.

```
[root@demo ~]# rpm --whatrequires gimp
gimp-help-2-0.1.0.10.1.1
gimp-data-extras-2.0.1-1.1.1
gutenprint-plugin-5.0.1-3.fc7
xsane-gimp-0.991-5.el5
gimp-lqr-plugin-0.4.0.4-2.fc7
```

This is the list of other packages that depend on the capabilities provided by this package.

What Provides a Given Capability?:

You can use the “--whatprovides” switch to find out which package (if any) will satisfy a given dependency:

```
[root@demo ~]# rpm -q --whatprovides 'rtld(GNU_HASH)'
glibc-2.5-24.el5_2.2
```

You can use the “--provides” switch to list the capabilities that a given package provides:

```
[root@demo ~]# rpm -q --provides bash
config(bash) = 3.2-21.el5
bash = 3.2-21.el5
```

Ignoring Dependencies:

It doesn't happen often, but sometimes it's necessary to forcibly install a package, even when the package's dependencies aren't satisfied. This may happen for several reasons:

• You're trying to install a malformed package. Sometimes mistakes are made when a package is created.

• You've installed something by hand, rather than through the “rpm” command. The missing dependencies may actually be satisfied by files you've installed by hand, but rpm doesn't know about this.

• The dependencies are actually satisfied, but the package(s) that satisfy them don't provide the expected “capabilities”. This can be either because you've mixed packages from different distributions, or because one of the already-installed packages is malformed.

In these cases, you can use the “--nodeps” switch to cause rpm to ignore apparent dependency problems.

Verifying and Re-installing Packages:

You can check the integrity of a package's installed files with the “-V” switch:

```
[root@demo ~]# rpm -V firefox
```

Files that have changed since installation will be displayed along with flags indicating how they have changed:

```
S file Size differs
M Mode differs (includes permissions and file type)
5 MD5 sum differs
D Device major/minor number mismatch
L readLink(2) path mismatch
U User ownership differs
G Group ownership differs
T mTime differs
```

If you think a package has been damaged, and you want to re-install it, you can use the “--replacepkgs” switch. Without this, rpm would just tell you that the package was already installed.

```
[root@demo ~]# rpm -i --replacepkgs \
firefox-3.0.6-1.el5.centos.i386.rpm
```

Repairing Broken RPM Databases:

The RPM database is stored in several files in the directory /var/lib/rpm. It's possible for this database to become corrupted. Most commonly, the database's index files are mangled. These are files in /var/lib/rpm whose names begin with "__db".

```
[root@demo ~]# ls -l /var/lib/rpm
-rw-r--r--  1 rpm  rpm   21475328 Mar 17 14:29 Basenames
-rw-r--r--  1 rpm  rpm      12288 Mar  5 12:46 Conflictname
-rw-r--r--  1 root root         0 Jan  9 01:51 __db.000
-rw-r--r--  1 root root     24576 Dec 26 10:23 __db.001
-rw-r--r--  1 root root   1318912 Dec 26 10:23 __db.002
-rw-r--r--  1 root root    450560 Dec 26 10:23 __db.003
-rw-r--r--  1 rpm  rpm    7958528 Mar 17 14:29 Dirnames
-rw-r--r--  1 rpm  rpm   21196800 Mar 17 14:29 Filemd5s
-rw-r--r--  1 rpm  rpm      65536 Mar 17 14:29 Group
-rw-r--r--  1 rpm  rpm      77824 Mar 17 14:29 Installtid
-rw-r--r--  1 rpm  rpm     172032 Mar 17 14:29 Name
-rw-r--r--  1 rpm  rpm  127090688 Mar 17 14:29 Packages
...etc.
```

When this happens, you'll find that (for example) "rpm -q" queries hang forever. To fix this, the index files can be deleted and rebuilt. Just delete the __db.* files, then type:

```
[root@demo ~]# rpm --rebuilddb
```

Note also that the "rebuilddb" process can take a long time, possibly hours.

Part 4: Automated Package Management with yum

But where to you find all of these packages so you can install them? And what about the pain of finding all of the other packages that a given package depends on? To solve those problems, there are higher-level package managment systems like “yum”.

Yum's name comes from “Yellow Dog Updater, Modified”. It's a modified version of a program originally found in the Yellow Dog Linux distribution, which is a Linux distribution for computers based on the Motorola PowerPC architecture.

The “yum” Command:

Yum is a frontend to rpm. It does several things:

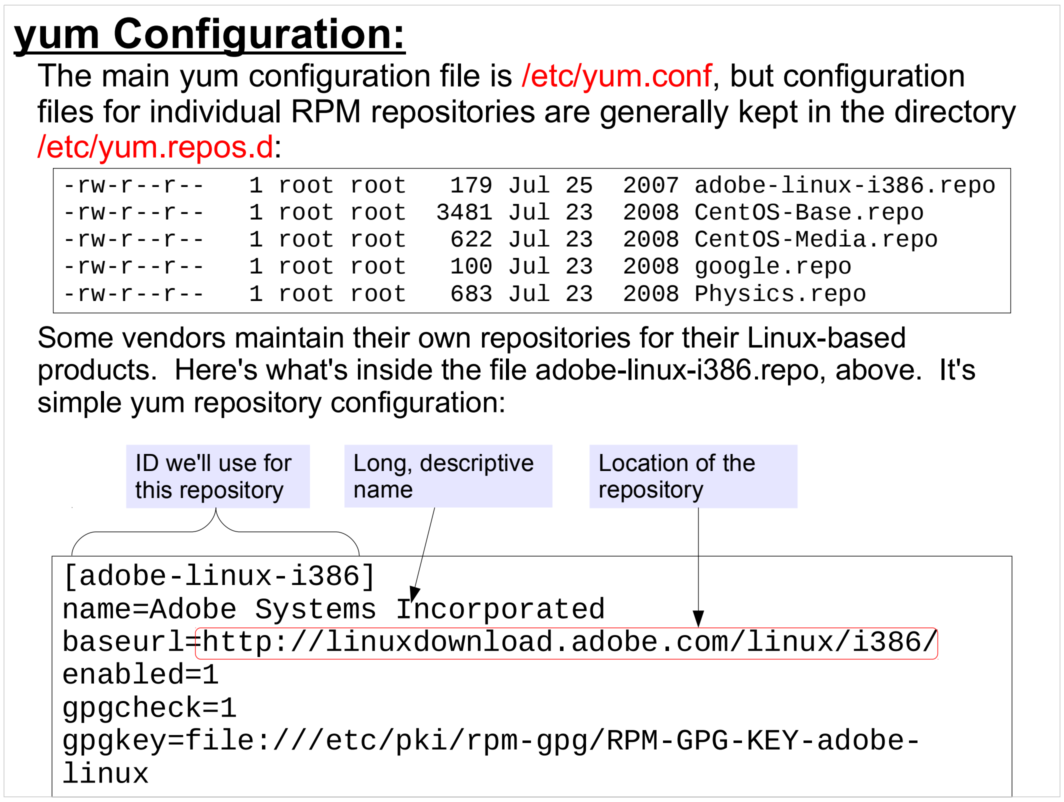
• Like rpm, it allows you to install, update, remove and query packages. It does this by intelligently executing rpm commands on your behalf.

• If you don't have the RPM file you need, yum will automatically fetch it (or try to) from a set of pre-defined “repositories”. These are locations on the internet where collections of RPM files are stored.

• If the package you want to install requires other packages (dependencies), yum will fetch those too, and also install them.

Installing a new package with yum is often a simple as:

```
[root@demo ~]# yum install firefox
```



yum Configuration:

The main yum configuration file is /etc/yum.conf, but configuration files for individual RPM repositories are generally kept in the directory /etc/yum.repos.d:

```
-rw-r--r--   1 root root   179 Jul 25  2007 adobe-linux-i386.repo
-rw-r--r--   1 root root  3481 Jul 23  2008 CentOS-Base.repo
-rw-r--r--   1 root root   622 Jul 23  2008 CentOS-Media.repo
-rw-r--r--   1 root root   100 Jul 23  2008 google.repo
-rw-r--r--   1 root root   683 Jul 23  2008 Physics.repo
```

Some vendors maintain their own repositories for their Linux-based products. Here's what's inside the file adobe-linux-i386.repo, above. It's simple yum repository configuration:

```
[adobe-linux-i386]
name=Adobe Systems Incorporated
baseurl=http://linuxdownload.adobe.com/linux/i386/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-adobe-
linux
```

A yum repository is just a directory full of RPMs, with another directory, called “repodata” inside it. The repodata directory contains index information about the RPM files. You can create a yum repository by pointing the command “createrepo” at a directory full of RPMs.

Updating with yum:

Yum offers you a simple way to update all of your packages to the latest versions found in your collection of repositories. Just type:

```
[root@demo ~]# yum update
```

Yum will then scan the repositories for updated versions of all of your installed packages. Yum will give you a list of any updates it finds, and ask you if it's OK to install them. If you want to skip this confirmation, just add the "-y" switch ("yum -y update").

If you just want to get a list of the available updates, type:

```
[root@demo ~]# yum check-update
```

To update just one package, append its name:

```
[root@demo ~]# yum update firefox
```

More yum Commands:

The following command would remove the “firefox” package:

```
[root@demo ~]# yum remove firefox
```

This will list all currently-installed packages, just like “rpm -q -a”:

```
[root@demo ~]# yum list installed
```

This will list packages available in the known repositories, but not yet installed on your computer:

```
[root@demo ~]# yum list available
```

Automatic Updates with yum:

The package called “yum-cron” will schedule automatic updates for your computer, once per day. It uses the “cron” service, which we'll discuss in a later talk.

To install and enable it, just type:

```
[root@demo ~]# yum -y install yum-cron
[root@demo ~]# chkconfig yum-cron on
[root@demo ~]# service yum-cron start
```

This package causes yum to be run once per day, at midnight, to automatically install any applicable updates.

Comparison of RPM and DEB Commands:

Package management in the Debian/Ubuntu world is done with different commands, but they're just analogous to the commands we've seen for the RPM-based world. Debian-based distributions use “deb” packages, instead of “rpm” packages.

RPM	DEB
`rpm -i`	`dpkg -i`
`rpm -e`	`dpkg -r`
`rpm -q -i`	`dpkg -s`
`rpm -q -l`	`dpkg -L`
`rpm -q -a`	`dpkg -l`
`rpm -U`	`dpkg -i`
`yum install`	`apt install`
`yum remove`	`apt remove`
`yum update`	`apt update; apt upgrade`

Here are some screenshots of the Ubuntu 18.04 installation process:

http://www.ubuntubuzz.com/2018/03/how-to-install-ubuntu-1804-lts-bionic-beaver.html

Comparison of RPM and DEB Commands (cont'd):

RPM	DEB
rpm -q -l -p	dpkg –contents
rpm –rebuilddb	dpkg –configure -a
yum list...	apt-cache pkgnames (no “installed/available” option)
yum -y install	apt install
yum-cron	unattended-upgrades (see https://help.ubuntu.com/lts/serverguide/automatic-updates.html.en for more information.)

Comparison of RPM and DEB Commands (cont'd):

RPM	DEB
`rpm –requires`	`apt-cache depends`
`rpm –whatrequires`	`apt-cache rdepends`
`rpm –whatprovides`	`dpkg -S`
`/etc/yum.repos.d`	`/etc/apt/sources.list.d`

For more information about how to use the Debian package management system, see:

http://newbiedoc.sourceforge.net/tutorials/apt-get-intro/info.html

Thanks!