System Administration

Application Management

Application Management

- A common task for a system administrator is the installation, updating and removal of software. Several ideas exist to make that an easier job

Application Management

- Most linux distributions have some sort of package manager, which allows for easier installation, removal and updating of software.
- They also typically have some sort of dependency checking, allowing administrators to know about dependencies between packages, and (with some package managers) to handle automatically those dependancies

./configure; make; make install

- The traditional approach to UNIX application installation
- Application is shipped in a "tarball" file
- The tarball file is unpacked with the `tar` command, then `make` is used to compile and install the application

Tarball Demonstration

- Download the sample tarball from the website
- Execute the following commands:

```
tar -xzvf magicrescue-1.1.5.tar.gz
cd magicrescue-1.1.5
./configure
make
su
make install
```

Tarball Limitations

- How do I uninstall this package?
  Sometimes there's a `make uninstall`
- What packages are installed on my system?
- How do I keep those packages up to date?

The RedHat Package Manager

- RPM is the default package manager for Red Hat Linux systems.
- It deals with .rpm files, which contain the actual programs as well as various bits of meta-information about the package: what it is, where it came from, version information and info about package dependencies.

RPM Commands

- `rpm -ivh <filename(s)>`

  Installs (-i) packages. The -vh causes rpm to be verbose about what it is doing and to print hash marks (#) as it is installing packages

- `rpm -Uvh <filename(s)>`

  Updates (-U) packages.

RPM Demo

- Download the sample RPM file from the website
- Install it with the following two-line command (notice the continuation on the first line):

```
rpm -ivh checkinstall-1.6.0\
-3.el5.rf.i386.rpm
```

RPM Commands

- `rpm -q <package-name>`

  Queries information about a particular installed package.

- `rpm -qi <package-name>`

  Get more information: who packaged the rpm, where it came from, info about what it is

- `rpm -qip <filename>`

  Get information about an uninstalled .rpm file.

- `rpm -qR <package-name>`

  get package dependencies

RPM Commands

- `rpm -qf <filename>`

  Which package owns a file

- `rpm -q --filesbypkg <package>`

  Which files are installed in a package

RPM Commands

- You can combine the query commands:

  `rpm -qiRp --filesbypkg <filename>`

  Query the uninstalled rpm file <filename>, reporting information on the package, what dependencies it has and what files it will install.

Exercises

- Which package owns `/usr/bin/ssh`
- What other files come in that package?
- What dependencies does that package have?
- Who packaged that package?

RPM Shortcomings

- While RPM gives information about package dependencies, it doesn't do anything to help you resolve those dependencies: if package Foo requires package Bar, and package Bar requires package Baz and package Fnord (version 2.1 or later), rpm will tell you that information, but it won't help you install the dependencies.
- Someone has to package a program as an RPM. A piece of software you want may not be packaged.

yum
Yellow Dog Updater Modified

- Overcomes the first of RPM's shortcomings by locating and installing packages that meet dependencies
- Yum can work with a wide variety of software repositories (http, ftp, local files) and pull in the latest version of a package from all of them
- Yum also has plugins for special purposes (RHN)

Listing packages

- Listing installed packages
  `yum list installed [packageglob]`
- Listing packages available but not installed
  `yum list available [packageglob]`
- List available updates
  `yum list updates`
- List packages recently added to repositories
  `yum list recent`

Installing and updating packages

- Install a new package from the repository
  `yum install packageglob`
- Update packages from the repository
  `yum update [packageglob]`

  If you leave off the package names, all packages needing updates will be updated
- Upgrade packages (including replacing obsolete packages – theoretically can upgrade version)
  `yum upgrade [packageglob]`

Cleaning up after yum

- Delete the cached packages
  `yum clean packages`
- Eliminate yum's header files
  `yum clean headers`
- Remove all of yum's files and force it to download new ones:
  `yum clean all`

yum configuration files

- /etc/yum.conf – main config file
- /etc/yum.repos.d, /etc/yum/repos.d – directories containing repository description files
  - Description files are named somereponame.repo
  - Contain information about how you reach the repository (http, ftp, etc.), whether to use GPG keys, and (if used) the GPG key location

Where's Red Hat Network?

- Red Hat Network support is implemented through a yum plug-in rather than as a repository (see /etc/yum/pluginconf.d).
- This allows each machine to see a unique repository with its own set of packages based on the channels you've selected for it or its group
- Also allows Red Hat to shut you off when you haven't paid your subscription fee

Pirut – graphical package manager

- Appears in Gnome Applications menu as “Add/Remove Software”
- Using either Browse, List or Search views:
  - Check packages to install
  - Uncheck packages to uninstall
  - Click “Apply”

RHEL 4 and earlier -- up2date

- `up2date [pkgname]` will install or update a particular package, handling dependencies (between repositories, if necessary)
- `up2date -u` will update every package on a system
- The `--dry-run` flag to any up2date command will show you what up2date would do before doing it

yum/up2date registration

- Yum and up2date require that you have registered your system with RH to do updates
- The RHEL Group on campus provides a bootstrap script that will register a system to use the campus proxy server
- `up2date-config` allows you to configure manually
- `rhn_register` will also register a system

yum/up2date and signed packages

- By default, yum and up2date require that packages be gpg signed and that you have the public key of the signer
- RHEL comes with the default Red Hat keys
- The RHEL Group bootstrap script installs keys for local packages
- You can also install keys manually

```
rpm -ivh public-keyfile
```

Why sign packages?

- This gives you some indication that the rpm you have downloaded was actually packaged by the person who claims to have packaged it
- That trust only goes as far as how rigorously you believe in the public keys

Exercises

- What command would you run to update ssh?
- What command would you run to update everything on your system?

APT – another option

- APT is used in Debian and Debian-derived distros (like Ubuntu)

  `apt-get update` [updates apt database]
  `apt-get install` *`packagename`*
  `apt-get remove` *`packagename`*
  `apt-get -u upgrade` [upgrades all packages]

- Some distros call apt-get *aptitude*
- There is also a graphical front-end for APT called Synaptic

Building RPM Files

- The good way: `rpmbuild`
  - Requires the creation of a spec file containing information about the package
- The lazy way: `checkinstall`
  - Builds directly from a generic tarball
  - Doesn't have all of the information of an RPM built from a spec file, but it's simple and quick

checkinstall Demo

- Back to the magicrescue directory:

```
cd ~/magicrescue-1.1.5
/usr/local/sbin/checkinstall
```

  [answer the prompts...]

```
rpm -i --nodeps /usr/src/redhat/\
RPMS/i386/magicrescue-1.1.5-1.i386.rpm
```

- Verify the installation with

```
rpm -qi magicrescue
man magicrescue
/usr/local/bin/magicrescue
```