Applying OS Updates

1 Aim

After successfully working through this exercise, You will:

- understand the importance of applying updates to the operating system of servers
- understand how to apply the updates to Red Hat Linux.
- understand how to use RPM and `yum` to resolve software package dependencies
- understand how to automate the update process

2 Background

All operating systems contain bugs. Even Microsoft's OS, despite what Bill may say (See `http://www.cantrip.org/nobugs.html`), contains bugs. See for example,

`http://www.gcn.com/archives/gcn/1998/july13/cov2.htm`. And Linux contains bugs. Now crackers are just waiting for a new bug to be uncovered. The company that provides the OS also will provide updates to their OS, thanks to CERT: `http://www.cert.org/`, who eventually publish information about the bug even if the company does not. As a system administrator, it is a vitally important part of your duty to apply updates to important or sensitive computers in your company.

Here in the ICT Department, I have implemented a system that automatically downloads the current updates for Red Hat Linux from the Internet every day. You will use these updates and apply them to your computer. The updates are software packages.

You will use the Network File System (NFS) to *mount* the network drive containing these updates, and then use the RPM Package Manager (RPM) to apply these updates to your machine.

The RPM package manager does far more than allow you to install updates. It can also check that the installed software packages are correctly installed, and verify that none of the programs have been changed by a cracker (as long as RPM itself is unchanged!) See chapter 25, *Package Management with RPM*, of the *The Official Red Hat Linux Customization Guide*, available from the Red Hat web site at

`http://www.redhat.com/docs/manuals/linux/`, and also on the documentation CDROM (burn a copy on our CD burner).

2.1 The RPM Package Manager

The RPM package manager has a number of basic capabilities: It can *install* software, *update* software packages, *remove* packages, *build* software packages, *query* and *verify* software packages.

Please see section 6.10 on page 185 in the *Linux Training Materials Project* (the workshop notes we have used in the lab) for more information about using RPM to query and verify software packages.

Building RPM Packages If anyone is interested in building your own RPM packages, you are very welcome to see Nick, he may even give you a book about it! The online book is called *Maximum RPM*, and is good reading for those who are interested.

If you find an RPM package for another version of Linux, it is a good idea to rebuild it, since that will solve library dependencies. For this, you use the *source* RPM package. To do this is simple. For example, suppose you download the latest Emacs package from `ftp://rawhide.redhat.com/pub/redhat/linux/rawhide/SRPMS/SRPMS/emacs-21.1-3.src.rpm` and want to use it in Red Hat 9 instead of the older version there. All you need do is:

```
$ sudo rpmbuild --rebuild emacs-21.3-6.src.rpm
```

The code compiles, and eventually creates the binary RPMs, which you can then install:

```
$ sudo rpm -Uhv /usr/src/redhat/RPMS/i386/emacs-*21.3-6.i386.rpm
```

Table 1 on the next page shows the basic RPM operations that we will use today. Table 2 on page 4 shows more information about how to make queries using RPM.

3 Procedure

Note that updating the kernel requires that you keep the current kernel, otherwise the loadable modules will no longer be available for the currently running kernel.

We will use two methods; a manual method in which you manually resolve software package dependencies, and an automatic method using `yum`. The `yum` program is available from `http://linux.duke.edu/projects/yum/`.

There are three steps in the manual method:

- Access the network drive
- If there are kernel updates that need to be applied, apply them first, installing any dependent packages first.
- Apply the remaining updates, first those specialised to the Pentium II and above, and secondly, the remaining updates.

Here is how we do it.

3.1 Accessing the network drive

1. Type:

```
$ cd /home/nfs/redhat-9/updates
```

Now you have *automounted* the directory `/var/ftp/pub` from the machine `ictlab` over the network to the local directory `/home/nfs`, using the NFS protocol. When you change to your local directory `/home/nfs`, you will be accessing the directory `/var/ftp/pub` on `ictlab` over the network.

We say that `ictlab` *exports* the directory `/var/ftp/pub` by NFS, and that your machine has *mounted* this on the local directory `/home/nfs`.

Table 1: The RPM options that we use to install and upgrade software.

Command	*Action*	*Description*
`rpm -i`	install	install the software package; do not uninstall any previous version of the software. Use this when installing kernels; normally, use the `upgrade` action.
`rpm -U`	upgrade	if an older copy of the package has been installed, uninstall it and replace it with this newer package. Do not delete the configuration files from the old package. Otherwise, it is the same as the `install` action, and is what I use most often.
`rpm -F`	freshen	install this package only if an older version of this software package has already been installed. Otherwise, do nothing. Useful when upgrading software packages.
`rpm -e`	remove	will completely erase the software package.
		Useful “helper” options
`-h`	hashes	Print fifty hashes (‘#’) to show progress of installation
`-v`	verbose	print the name of each package as it is installed. Useful when installing more than one package at a time. In practice, I always use this option, as well as the `-h` option.
`--force`	force :-)	For updates, go ahead and force the update, even if the same package or a newer one is already installed. Be very careful in using this; understand what you are doing. It can also override dependencies, and you can end up with a system that does not work if you use this option carelessly.

3.2 The Organisation of the Updates

The updates, as organised on Red Hat’s ftp sites and mirrors, is in a group of directories.

1. Type

```
$ ls
athlon  i386  i586  i686  noarch  SRPMS
```

   Most binary packages we need are in the `i386` directory. Some platforms benefit from packages that are compiled specifically for their architecture, so if you have a P-III or P-4, install the packages from the `i686` directory rather than the same packages from the `i386` directory. Table 3 on the next page summarises what is in each of the directories.

4 Installing yum

1. Change to the directory on our server, and list all the yum software packages:

```
$ cd /home/nfs/redhat/contrib
$ ls yum*
```

Table 2: This is a brief list of RPM query commands. I have used the `openssh` package as an example.

command	*effect*
`rpm -qa | less`	list all installed software packages
`rpm -q openssh`	show the version of the `openssh` package,[a] if it is installed. Also use this to determine if a package is installed or not.
`rpm -qa | grep openssh`	show all installed packages that have *openssh* in their name
`rpm -ql openssh`	list all files in the openssh package
`rpm -qd openssh`	list all documentation files in the openssh package
`rpm -qc openssh`	list all configuration files in the openssh package
`rpm -qi openssh`	display information about the package
`rpm -V openssh`	verify that the openssh package is correctly installed
`rpm -qf /etc/passwd`	determine which package the `/etc/passwd` file belongs to

Table 3: The directories in the updates directory.

directory	*What it's for*
`athlon`	Contains packages for AMD's Athlon and Duron processors
`i386`	Most of the packages you install come from here
`i586`	Packages optimised for the Pentium
`i686`	Packages optimised for the Pentium-II
`noarch`	These software packages contain no architecture specific code, and are used on all platforms
`SRPMS`	The source RPM packages

2. Then install the latest:

```
$ sudo rpm -Uhv yum-2.0.3-1.noarch.rpm
```

5 Setting up yum

1. edit the main configuration file:

```
$ xhost +localhost
$ sudo -v
$ sudo emacs /etc/yum.conf &
```

- The first command allows the `root` user to display graphical output on your X display
- The second command allows you to type your password, and start a new five-minute period during which you can execute programs using `sudo` without typing a password

- The last command starts a *text editor* called `emacs`. The `&` at the end of the line executes the program in the *background*, so that you can type more commands while `emacs` is running

2. Copy the two sections near the bottom and duplicate them like this. To copy and paste, select the text with the mouse, then move the mouse to the end of the file, then press the middle mouse button (or the wheel if a wheel mouse, or both mouse buttons at the same time if it is a two button mouse.)

```
[base]
name=Red Hat Linux $releasever - $basearch - Base
baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/$releasever/$basearch/

[updates]
name=Red Hat Linux $releasever - Updates
baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/updates/$releasever/

[base]
name=Red Hat Linux $releasever - $basearch - Base
baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/$releasever/$basearch/

[updates]
name=Red Hat Linux $releasever - Updates
baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/updates/$releasever/
```

3. Put *comment characters* (the '`#`' character) at the start of the original two lines, then change the copies, so that the end of the file looks like this:

```
# [base]
# name=Red Hat Linux $releasever - $basearch - Base
# baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/$releasever/$basearch/

# [updates]
# name=Red Hat Linux $releasever - Updates
# baseurl=http://mirror.dulug.duke.edu/pub/yum-repository/redhat/updates/$releasever/

[base]
name=Red Hat Linux $releasever - $basearch - Base
baseurl=file:///home/nfs/redhat-$releasever/RedHat/

[updates]
name=Red Hat Linux $releasever - Updates ictlab
baseurl=file:///home/nfs/redhat-$releasever/updates/
```

4. Now enable the yum service:

```
$ sudo /sbin/chkconfig yum on
$ /sbin/chkconfig yum --list
yum             0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

See the lectures notes about *runlevels*. The lecture note slides discuss what you are doing here in detail. Try the exercises described in the lecture notes.

5. Now run it once, manually:

```
$ sudo yum update
```

5.1 Using `yum` Elsewhere

Here we changed the `/etc/yum.conf` configuration file so that your machine would download the updates from our local server `ictlab`. It is more efficient for all of us to download them once, then have all machines update from our local server. The alternative, to have each machine update directly from the machine at Duke University in the United States would be less efficient, and less friendly with network resources.

However, at home, or for a few machines at work, you should simply install `yum` and let it download the updates from Duke University. You would not need to edit the `/etc/yum.conf` file at all.

Note that there are many other `yum` repositories listed at the `yum` website. The most important are the fedora web sites, since the fedora stable packages are sponsored by Red Hat, and freshrpms, which provides software packages for many things, including reading and writing DVDs and for playing MP3s with Xmms.