RedHat Linux KickStart HOWTO

Table of Contents

RedHat Linux KickStart HOWTO

Martin Hamilton <martinh@gnu.org>

v0.2, 11 January 1999

This HOWTO briefly describes how to use the RedHat Linux KickStart system to rapidly install large numbers of identical Linux boxes. For advanced users, we describe how to modify the KickStart installation procedure to do your own thing, and give a quick guide to building RPM packages of your own.

7.Installation itself

8.Mounting the boot/supp disks

9.Modifying the RedHat installer

10.FAQs/Wish list

11.Credits

12.Appendix A – Configuring BOOTP/DHCP and NFS

13.Appendix B – Making your own RPMs

14.Appendix C – Munging your own RPMs into the distribution

1.Copyright

2.Homepage

If you got this document from a Linux HOWTO mirror site or a CD−ROM, you might want to check back to the KickStart HOWTO home page to see if there's a newer version around.

3.Introduction

RedHat Linux version 5 comes with a a little−known (and until now, not hugely documented) feature called *KickStart*. This lets you automate most/all of a RedHat Linux installation, including:

- Language selection
- Network configuration and distribution source selection
- Keyboard selection
- Boot loader installation (e.g. lilo)
- Disk partitioning and filesystem creation
- Mouse selection
- X Window system server configuration
- Timezone selection
- Selection of an (initial) root password
- Which packages to install

Eagle eyed RedHat users will probably have realised by now that these are essentially the main steps involved in the manual installation of a RedHat Linux system. KickStart makes it possible for you to script the regular installation process, by putting the information you would normally type at the keyboard into a configuration file.

But wait − there's more :−)

Having finished the normal installation process, KickStart also lets you specify a list of shell level commands which you would like to be executed. This means that you can automatically install extra local software not distributed as part of RedHat Linux (yes, there are even more free software programs than the ones you get with the RedHat distribution. Some can't be distributed by RedHat on legal grounds, e.g. the `ssh` and `PGP` encryption systems) and carry out any tidying up you may need to do in order to get a fully operational system.

4.Prerequisites

There are two approaches to a KickStart install – one is to simply copy your KickStart configuration file to a RedHat boot floppy. The other is to use a regular boot floppy and get your KickStart config file off the network.

In both cases, you'll need:

1. Intel (i386) class machines – KickStart appears to only work on these at the time of writing.
2. KickStart config file – we'll talk about this in the next section.
3. RedHat boot disk – preferably from the *updates* directory, to take advantage of any fixes/driver updates.
4. DNS entries for the IP addresses you'll be using – optional, but will stop the installation from prompting you for your machine's domain name.

If you want to fetch your config file over the network, you'll need to export it via NFS – this is the only access method supported at the moment. The config file lets you specify a different NFS server to fetch the RedHat distribution itself from.

You can configure a static IP address for your machine – e.g. a special one reserved for KickStart installations. Alternatively, if you don't want to hard code an IP address in the config file you can tell KickStart to use a BOOTP/DHCP server to fetch this. Some servers will allocate new addresses in a given range automatically, e.g. the CMU BOOTP server with dynamic addressing extensions.

More information on NFS and BOOTP/DHCP is in Appendix A.

5.Setting up a boot floppy

Essentially, all you have to do is copy your KickStart config file to */ks.cfg* on the RedHat boot floppy, e.g.

```
mcopy ks.cfg a:
```

However – the RedHat boot floppy is pretty full, and you may find that you have to delete some of the other files to make room for the KickStart config file. I was able to make enough room for mine by deleting the various message files normally displayed by the `SYSLINUX` boot loader, e.g.

```
mdel a:\*.msg
```

Another approach would be to throw away the drivers for some of the hardware you don't have − see the section on modifying the boot floppy below.

You may also want to edit *syslinux.cfg*, the `SYSLINUX` config file. This also lives in the top level directory of the RedHat boot floppy. For instance, the following *syslinux.cfg* will cause KickStart mode to be entered into automatically as the machine boots up, without the normal delay:

```
default ks
prompt 0
label ks
  kernel vmlinuz
  append ks=floppy initrd=initrd.img
```

Note that you almost probably want to base your boot and supplementary floppies on the most recent disk images available in the *updates/i386* on your local RedHat mirror site. Older images may be buggy or have driver support for less hardware.

6.The KickStart config file

There are three main sections to the config file:

1. System info, e.g. disk partitioning and network config
2. RedHat packages to install
3. Post−installation shell commands to run

There are some other possibilities which we won't talk about here, but **might** work. For more information check out the sample KickStart config in *misc/src/install/ks.samp* and *doc/README.ks* under the top level *i386* RedHat distribution directory on your CD−ROM or local RedHat mirror site.

6.1 System info

The available directives which I've been using are:

lang

Language configuration, e.g. for English
`lang en`

network

Network configuration, e.g. to use BOOTP/DHCP
`network --bootp`

nfs

NFS server and directory to install from, e.g.
`nfs --server chicken.swedish-chef.org /mnt/cdrom` to use the NFS server *chicken.swedish−chef.org* and try to mount the RedHat distribution from the directory */mnt/cdrom*.

keyboard

Select keyboard type, e.g. for UK keyboards
`keyboard uk`

zerombr

Clear the Master Boot Record − removes any existing operating system boot loader from your disk

clearpart

Clear existing partitions − e.g. to remove all existing disk partitions prior to installation
`clearpart --all`

part

Partition the disk, e.g. to make a root filesystem of 500MB
`part / --size 500`

install

Make a fresh installation of RedHat Linux.

mouse

Set the mouse being used, e.g. for a PS/2 or compatible "bus mouse"

```
mouse ps/2
```

timezone

Set the timezone, e.g. for local time in the UK

```
timezone --utc Europe/London
```

rootpw

Set the initial root password, based on a previously derived encrypted password

```
rootpw --iscrypted XaacoeGPmf/A.
```

lilo

Install the LILO boot loader, e.g. in the Master Boot Record

```
lilo --location mbr
```

%packages

Packages to install – see below.

%post

Post–installation shell commands – see below.

Note that the directory where KickStart is looking for the RedHat distribution should have a subdirectory *RedHat*, which contains the RedHat distribution tree for the platform in question. In the above example, we should see something like the following files and directories:

```
/mnt/cdrom/RedHat
/mnt/cdrom/RedHat/base
/mnt/cdrom/RedHat/contents
/mnt/cdrom/RedHat/i386
/mnt/cdrom/RedHat/instimage
/mnt/cdrom/RedHat/RPMS
/mnt/cdrom/RPM-PGP-KEY
```

If you're installing off a CD–ROM rather than off the network, the contents should look something like this:

```
RedHat
RedHat/base
RedHat/contents
RedHat/i386
RedHat/instimage
RedHat/RPMS
RPM-PGP-KEY
```

If you have the RedHat distribution for multiple architectures (e.g. on an NFS server – they're too big to fit more than one architecture's version onto a single CD–ROM), you'll notice that each distribution has the same files and directories under a subdirectory, e.g.

```
alpha/RPM-PGP-KEY
i386/RPM-PGP-KEY
sparc/RPM-PGP-KEY
```

There should be a file `architecture`/Redhat/`architecture`, e.g. *i386/Redhat/i386*.

If you want to create your own encrypted passwords, it's very easy using Perl, e.g.

```
% perl -e 'print crypt("schmurrdegurr", "Xa") . "\n";'p
```

Other options (or mooted options), which I've not tried:

cdrom

Install off CD–ROM rather than network.

device

Explicitly declare device details, e.g. `device ethernet 3c509 --opts "io=0x330, irq=7"` Alternative values of `device` include `scsi` for SCSI controllers and `cdrom` for proprietary CD–ROM drives.

upgrade

Upgrade an existing installation rather than make a fresh installation.

xconfig

Configure X Window server, graphics card and monitor. e.g.

```
xconfig --server "Mach64" --monitor "tatung cm14uhe"
```

I've not delved too deeply into this last one, because I'm not ever planning to run X on the console of any of my KickStarted machines. I'm told that running `xconfig` within KickStart itself is a bit flaky, but the same functionality is also available from the command line via `Xconfigurator` – so you might be best off leaving this to the post–installation script.

Here's how this first part of a KickStart config file looks when we put all the bits together:

```
lang en
network --static --ip 198.168.254.253 --netmask 255.255.255.0
  --gateway 198.168.254.1 --nameserver 198.168.254.2
nfs --server chicken.swedish-chef.org /mnt/cdrom
keyboard uk
zerombr yes
clearpart --all
part / --size 500
part swap --size 120
install
mouse ps/2
timezone --utc Europe/London
rootpw --iscrypted XaacoeGPmf/A.
lilo --location mbr
```

Note that some of the RedHat documentation refers to an invocation of the `network` directive which doesn't actually work in practice: `network --option`. The correct invocation is to put `network` followed by `--static`, `--bootp` or `--dhcp`. Be aware that the BOOTP and DHCP options are different – to the extent that they even use different code.

You can add the `--grow` parameter to a `part` directive to indicate that it's OK to grow the partition beyond the size you specify. It probably only makes sense to have one partition tagged with `--grow`.

6.2 Packages to install

The start of the packages section of the KickStart config file is indicated by the presence of a `%packages` directive on a line of its own. This is followed by one or both of two types of package specifier – individual packages may be installed by giving the name of their RPM (excluding the version and platform information), and groups of packages may be installed by giving their group name.

Here's a sample `packages` section for a KickStart config file:

```
%packages
@ Base
netkit-base
bind-utils
ncftp
rdate
tcp_wrappers
traceroute
cmu-snmp
```

So, what are these groups ? Well, there are a number of groups defined by default in a file called *base/comps* under the RedHat distribution's top level directory. Here are the ones which were current at the time of writing:

- Base
- Printer Support
- X Window System
- Mail/WWW/News Tools
- DOS/Windows Connectivity
- File Managers
- Graphics Manipulation
- X Games
- Console Games
- X multimedia support
- Console Multimedia
- Print Server
- Networked Workstation
- Dialup Workstation
- News Server
- NFS Server
- SMB (Samba) Connectivity
- IPX/Netware(tm) Connectivity
- Anonymous FTP/Gopher Server
- Web Server
- DNS Name Server
- Postgres (SQL) Server
- Network Management Workstation
- TeX Document Formatting
- Emacs
- Emacs with X windows
- C Development
- Development Libraries
- C++ Development
- X Development
- Extra Documentation

You'll notice that they correspond to the various configurations which you're prompted for during a manual installation. Note that some of the packages in a given package group are duplicated in other groups, and that you can install multiple groups of packages without this causing problems. Each group's entry in the *comps* listing looks similar to this:

```
0 Extra Documentation
sag
lpg
howto
faq
man-pages
end
```

It seems that groups with a *1* next to their name (the first line above) are selected for installation by default. You can customise the Linux installation process even further by creating your own groups or redefine existing ones by editing this file.

6.3 Post−installation shell commands

This is probably the best feature of all, and something which there is no direct equivalent to in the manual installation process. What we can do here is specify a sequence of shell level commands which should be executed after the main installation (disk partitioning, package installation, and so on) is complete.

The beginning of this section is signified by the `%post` directive in the KickStart config file. In what follows you can take advantage of all of the utilities which have been installed on your newly built Linux system, e.g.

```
%post
ln -s /etc/rc.d/init.d /etc/init.d
ln -s /etc/rc.d/rc.local /etc/rc.local
ln -s /usr/bin/md5sum /usr/bin/md5
ln -s /usr/bin/perl /usr/local/bin/perl
chmod ug-s /bin/linuxconf
mkdir /var/tmp/tmp
perl -spi -e 's!image=/boot/vmlinuz-.*!image=/boot/vmlinuz!' /etc/lilo.conf
rm /etc/rc.d/rc*.d/*sendmail
```

You can also use I/O redirection and here documents:

```
cat <<EOF >>/etc/passwd
squid:*:102:3500:Squid Proxy:/usr/squid:/bin/bash
EOF

cat <<EOF >>/etc/group
cache:x:3500:
EOF
```

Modify the run−time startup scripts:

```
cat <<EOF >>/etc/rc.local
echo 8192 > /proc/sys/kernel/file-max
echo 32768 > /proc/sys/kernel/inode-max

[ -x /usr/sbin/sshd ] && /usr/sbin/sshd
[ -x /usr/sbin/cfd ] && /usr/sbin/cfd

EOF
```

Set up *crontab* entries:

```
cat <<EOF >/tmp/crontab.root
# Keep the time up to date
0,15,30,45 * * * * /usr/sbin/ntpdate -s eggtimer 2>&1 >/dev/null
# Recycle Exim log files
1 0 * * * /usr/exim/bin/exicyclog
# Flush the Exim queue
0,15,30,45 * * * * /usr/exim/bin/exim -q
EOF

crontab /tmp/crontab.root
rm /tmp/crontab.root
```

And even install other RPMs which you made yourself:

```
rpm -i ftp://chicken.swedish-chef.org/rpms/squid.rpm
rpm -i ftp://chicken.swedish-chef.org/rpms/ssh.rpm
rpm -i ftp://chicken.swedish-chef.org/rpms/exim.rpm
rpm -i ftp://chicken.swedish-chef.org/rpms/cfengine.rpm
rpm -i ftp://chicken.swedish-chef.org/rpms/linux.rpm

ssh-keygen -b 1024 -f /etc/ssh_host_key -N ""
```

```
depmod -a
```

Note that you can achieve the same effect by making your own RPMs containing the commands you want executed – see below for more information. Give them a carefully chosen name and you can force them to be installed first (e.g. name starts with 'aaa') or last (e.g. name starts with 'zzz').

Be aware that a less painful way of doing root crontab entries is to create them as files in one or more of the directories */etc/cron.hourly*, */etc/cron.daily*, */etc/cron.weekly* and */etc/cron.monthly*.

More information about making your own RPMs is available in Appendix B.

7.Installation itself

Boot the to–be–installed machine off your RedHat boot floppy as usual, but instead of pressing `RETURN` at the `SYSLINUX` prompt, type `linux ks`.

If you're lucky, this will be all you have to type!

If you customised your RedHat boot floppy as outlined above, you won't even need to do this bit :–)

Since we're really just automating the normal steps involved in a RedHat installation, the normal dialogs may appear if/when KickStart gets confused about what to do next. The most likely case is that your network interface won't be detected automatically, and you'll be prompted for its IRQ and I/O address space. KickStart tends to need help for ISA bus cards, but detects PCI bus cards automatically.

You can keep an eye on what KickStart is doing by by switching virtual consoles as usual:

- Alt–F1 – installation dialog
- Alt–F2 – shell prompt
- Alt–F3 – install log (messages from install program)
- Alt–F4 – system log (messages from kernel, etc.)
- Alt–F5 – other messages

8.Mounting the boot/supp disks

The RedHat boot disk *boot.img* is in MS−DOS format, using the `SYSLINUX` program to boot up. The supplementary disk *supp.img* is a Linux `ext2` filesystem. If you have support for the loopback filesystem in your Linux kernel, you can mount both of these files in your filesystem and hack at them:

```
# mkdir -p /mnt/boot /mnt/supp
# mount -o loop -t msdos boot.img /mnt/boot
# mount -o loop supp.img /mnt/supp
```

Now you should be able to see and manipulate the files on the boot and supplementary disk under */mnt/boot* and */mnt/supp* respectively. Phew! Note that older versions of `mount` may not be able to handle the `-o loop` option. In these cases you'll need to explicitly use `losetup` to configure the loopback device for each file, e.g.

```
# losetup /dev/loop0 boot.img
# mount -t msdos /dev/loop0 /mnt/boot
```

You might also need to explicitly use the `-t ext2` option when mounting an `ext2` filesystem like the one on the supplementary disk. But, it looks like people with modern Linux distributions shouldn't have to worry about this.

Of course, if you don't want to mess around too much, you can cut a corner and manipulate actual floppy disks rather than these floppy disk images. If time is important, you'll probably prefer to use the loopback devices, since you can hack around with the disk images without incurring the latency associated with a genuine floppy disk read/write.

9.Modifying the RedHat installer

If you want to mess around with the installation procedure itself, the source code can be found on the RedHat CD−ROM or your local RedHat mirror site. It's in *misc/src/install* under the *i386* distribution top level directory.

If you examine the RedHat boot disk you'll see that, in addition to the Linux kernel *vmlinuz*, there's a large file *initrd.img*:

```
- -rwxr-xr-x   1 root     root          559 May 11 15:48 boot.msg
- -rwxr-xr-x   1 root     root          668 May 11 15:48 expert.msg
- -rwxr-xr-x   1 root     root          986 May 11 15:48 general.msg
- -rwxr-xr-x   1 root     root       968842 May 11 15:48 initrd.img
- -rwxr-xr-x   1 root     root         1120 May 11 15:48 kickit.msg
- -r-xr-xr-x   1 root     root         5352 May 11 15:48 ldlinux.sys
- -rwxr-xr-x   1 root     root          875 May 11 15:48 param.msg
- -rwxr-xr-x   1 root     root         1239 May 11 15:48 rescue.msg
- -rwxr-xr-x   1 root     root          402 May 11 15:48 syslinux.cfg
- -rwxr-xr-x   1 root     root       444602 May 11 15:48 vmlinuz
```

You guessed it, this is another `ext2` filesystem saved as a file −− but with a twist. It's actually compressed as well. You can uncompress it and then mount the result, e.g.

```
# gzip -dc /mnt/boot/initrd.img >/tmp/initrd.ext2
# mkdir /mnt/initrd
# mount -o loop /tmp/initrd.ext2 /mnt/initrd
```

Probably the most important part of this filesystem is the collection of loadable kernel modules which are included with the boot disk. If you need to merge in a new version of a driver, you'll need to either replace *vmlinuz* with a new kernel which has this statically linked, or replace it in the modules collection. What's more, you may need to throw other modules away to make room.

The modules collection is the file *modules/modules.cgz*. Wondering what that might be ? It's actually a compressed `cpio` archive, believe it or not! And you thought nobody used `cpio` any more... Actually RPM itself uses `cpio` internally, too. Here's how to hack around with it:

```
# gzip -dc /mnt/initrd/modules/modules.cgz >/tmp/modules.cpio
```

```
# cpio -itv <modules.cpio >modules.listing
# mkdir modules
# cpio -idumv <../modules.cpio
```

I don't believe that there is currently a way under Linux (at least in mainstream distributions) to transparently access compressed filesystems. Let me know if you know better!

If you change anything, remember to:

1. Use `cpio` to recreate the archive. How to do this is left as an exercise for the reader...
2. Use `gzip` to compress the resulting archive.
3. Copy it to */mnt/initrd*, or wherever you put the uncompressed *initrd.img* archive.
4. Unmount */mnt/initrd* (or whatever you called it).
5. Compress the new *initrd.img* using `gzip` again.
6. Copy the resulting archive onto the boot disk image – */mnt/boot/initrd.img* in our example.
7. Unmount the boot disk image, e.g. */mnt/boot*.

Finally, you can now create new boot floppies using this modified boot disk setup, e.g.

```
# cat boot.img >/dev/fd0
```

10.FAQs/Wish list

Q: After KickStart installation, my machine won't boot up. The BIOS complains with a message like `Missing operating system`.

A: Sounds like the partition with the root filesystem on isn't bootable. Use `fdisk` to toggle its bootable status.

Q: After the floppy boots, I get the message: `Error opening files for kickstart copy: File exists`.

A: Use a more recent version of *boot.img* and *supp.img* – look in the *updates* directory of your local RedHat mirror site. There was a bug in some older versions of these for RedHat 5.1.

Q: Can you have all outstanding patches (update RPMs) applied automatically too ? How ?

A1: Copy the RPMs you want installing to the RPMS directory from which the installation is going to take place, get rid of the older RPMs, and update the file *RedHat/base/hdlist* with the new RPM details. See Appendix C for a script from Eric Doutreleau to do this for you. If you do this yourself, remember to run *genhdlist* afterwards!

A2: Try this Perl script: patchup. This compares the RPMs your system has installed with those in a nominated directory and reports on the ones it thinks need updating. It can even install them for you if you trust it to.

A3:rpm2hml has a much more powerful (12MB of C code vs. a page of Perl!) version of A2.

Q: A single config file on the install server for all of the clients, perhaps as a fallback after trying *IPADDR−kickstart* ?

A1: Use the BOOTP/DHCP 'boot file' parameter *bf* to set the filename.

A2: Add a a record `bf=/kickstart/ks.cfg` to the relevant entry in */etc/bootptab*.

Q: More flexibility when things go wrong – e.g. prompt for alternate locations if distribution not found on CD−ROM.

A: ?

Q: Explicit exclusion of packages – e.g. everything apart from *sendmail*.

A: Rebuild the **BASE** package without sendmail.

Q: Choose which services are started automatically on boot−up by the run−level scripts under */etc/rc.d/*.

A: The *chkconfig* utility lets you configure which services are run automatically on boot−up. You can run this in your post−installation script section, e.g. to run *ypbind* in run levels 3, 4 and 5:

```
chkconfig --level 345 ypbind on
```

and it will start the ypbind level on the 345 level.

Q: When executing the shell commands in the `%post` section, bring any output up in another virtual console rather than overwriting the main screen. *Could be done in the shell commands section using `open?`.*

A: No problem – do something like this:

```
exec >/dev/tty5
```

Q: Does the filesystem creation code check for bad blocks ?

A: If you switch to the virtual console where the filesystem creation output is being displayed, you won't see any mention of the 'read–only' test being performed. Looks like the answer is no.

Q: Can I arrange things so some of my machines are configured differently from others ?

A: You could move the host dependent stuff into the scripted section of the KickStart config – e.g. only install a given RPM if on a given machine. It would be useful to have a conditional installation feature in the packages section of the config file, e.g. switching on architecture, or hostname/domain name/IP address.

Q: Are there any changes between RedHat 5.1 and 5.2 ?

A1: Lots of changes in the installer, but mostly bug fixes or cosmetic improvements. No impact on KickStart as far as I can tell – from a *diff –rcs* of the two *misc/src/install* directories.

A2: RH5.2 now apparently includes the automatic IP allocation/DHCP patches to `bootpd`, but they have left out the documentation which tells you how to use it.

Q: (How) can you clear a specific partition or partitions ? e.g. to leave */home* but zap */*.

A: You can't – yet!

Q: Can you arrange to have your partitions created across multiple drives ? e.g. */* on `sda` and */home* on `sdb`.

A: Don't think so – looks like you only get access to the first drive from the partitioning tool.

Q: Is it possible to specify existing partitions to be included in the mount table, or is it only possible to specify the creation of new partitions that will then be included?

A: ?

Q: After running `mkkickstart`, where is the file it creates?

A: It doesn't create a file – it dumps the KickStart config to `stdout`.

Q: In virtual console 4 (Alt–F4) I get `Unable to load NLS charset cp437(nls_cp437).` What does this mean ? Should I be worried ?

A: Sounds like you're trying to mount a CD–ROM burned with the Joliet (Unicode extensions to ISO 9660. In theory the filenames on the CD–ROM might get munched and not make it through to Linux correctly. In practice it doesn't seem to cause any problems – could be a spurious dependency ?

Q: Why am i getting the X Window System installed ? I didn't put it in my list of packages.

A: The `XFree86-VGA16` RPM is a 'base' component, and as such always gets installed – unless you change the definition of the base class.

Q: In my post–installation script, can I use the packages which have been installed by now to do funky things not possible with the limited tools on the floppies ?

A: Yep – e.g. if you chose to install Perl when you put your KickStart config together, almost anything is possible in about five lines :–)

11.Credits

Thanks to Eric Doutreleau for the info about *chkconfig*, the `SYSLINUX` config file hack, and the Perl script for updating your distribution server's RPMs. Thanks to Robert Kaminsky for extensive investigations. Thanks to Piete Brooks, Flavia Regina Munhoz, Tom Toffoli, Bob Robbins, Charlie Brady and Ragen

Herrington, for their comments and questions.

12.Appendix A − Configuring BOOTP/DHCP and NFS

If you're wondering what on earth this BOOTP and DHCP stuff is, more information is available at the DHCP WWW site. NFS is documented separately in detail in the NFS HOWTO, and there's a DHCP mini−HOWTO too. I've tried to provide enough details here to help you get started, whilst not treating the topics in depth − let me know if you think this is overkill.

In the BOOTP/DHCP + NFS configuration we're discussing, the KickStart config file should be NFS mountable by the machine being installed from */kickstart/IPADDR−kickstart* on the BOOTP/DHCP server, where *IPADDR* is the IP address of the new machine, e.g. */kickstart/198.168.254.254−kickstart* for the machine *198.168.254.254*.

You should be able to override this location by returning the `bf` parameter (boot file) in your BOOTP/DHCP response. It may even be possible to have this NFS mounted off another machine entirely.

To NFS export some directories from an existing Linux box, create the file */etc/exports* with contents something like:

```
/kickstart *.swedish-chef.org(ro,no_root_squash)
/mnt/cdrom *.swedish-chef.org(ro,no_root_squash)
```

Note that if you didn't register the IP addresses you're going to be using in the DNS, you may be told to get lost by the NFS server and/or the RPC portmapper. In this you can probably get away with putting IP address/netmask pairs in the config files, e.g.

```
/kickstart 198.168.254.0/255.255.255.0(ro,no_root_squash)
```

and in */etc/hosts.allow*:

```
ALL: 194.82.103.0/255.255.255.0: ALLOW
```

This is because most Linux distributions use TCP wrappers to do access control for some or all of the NFS related daemons. Be aware that the */etc/exports* syntax tends to be different on other Unix variants – the NFS servers bundled with Linux distributions tend to offer a much wider range of options than the ones shipped with other versions of Unix.

Be aware that if you include a root password in your KickStart config file, or NFS export directories containing sensitive information, you should take care to expose this information to as few people as possible. This can be done by making the NFS export permissions as fine grained as you can, e.g. by specifying a particular host or subnet to export to rather than a whole domain. If you keep a special IP address free for KickStart installations, everything's nice and simple, but you'll have to change it later – or reconfigure the machine to get its IP address via BOOTP/DHCP.

Most NFS servers require you to tell `mountd` and `nfsd` (on some versions of Unix they're prefixed with a `rpc.`) that the */etc/exports* file has changed – usually by sending a `SIGHUP`. There's often a program or script called `exportfs`, which will do this for you, e.g.

```
# exportfs -a
```

If you didn't have NFS up and running when this machine was booted, the directories may not be exported automatically. Try rebooting, or running the following programs as root:

```
# portmap
# rpc.nfsd
# rpc.mountd
```

As noted, on some systems the `rpc.` prefix isn't used. In most modern Unix distributions, these programs can be found in the */usr/sbin* or */usr/libexec* directories. These might not be in your path already, e.g. if you used `su` to become *root*. The `portmap` program is also sometimes called `rpcbind`, e.g. on Solaris, some versions of `nfsd` require a command line argument specifying the number of instances of the server to run, and you may find you also need to run another daemon called `biod`. The above should suffice on most (all?) Linux systems.

If you're using the CMU BOOTP server with DHCP and dynamic addressing extensions referred to earlier, a sample */etc/bootptab* entry (*/etc/bootptab* is the normal location of the BOOTP/DHCP configuration file) would look something like this:

```
.dynamic-1:ip=198.168.254.128:T254=0x30:T250="ds=198.168.254.2:
dn=swedish-chef.org:sm=255.255.255.0:gw=198.168.254.1:
dl=0xFFFFFFFF":
```

(wrapped for clarity)

This says to allocate IP addresses dynamically on encountering new machines, starting at *198.168.254.128* and continuing for the next 48 (the hexadecimal value *30*) addresses. Each client will be passed back the value of *T250*. In this case that sets:

- the DNS server `ds` to *198.168.254.2*
- the domain name `dn` to *swedish−chef.org*
- the subnet mask `sm` to *255.255.255.0*
- the default gateway `gw` to *198.168.254.1*
- the lease length `dl` (how long the address is valid for) to "forever"

There seem to be a number of other versions of this server kicking around which do not support dynamic addressing. For these, you would have to list the hardware (typically Ethernet MAC) address of each to−be−installed machine in */etc/bootptab*, and the entries would look something like this:

```
bork.swedish-chef.org:ip=198.168.254.128:ha=0000E8188E56:
  ds=198.168.254.2:dn=swedish-chef.org:sm=255.255.255.0:
  gw=198.168.254.1:dl=0xFFFFFFFF":
```

(wrapped for clarity)

Note that the parameter `ha` corresponds to the hardware address of the machine being installed.

13.Appendix B − Making your own RPMs

The RPM package format is already very well documented, particularly in the book *Maximum RPM* by Ed Bailey, which you can download from the RPM WWW site − also available from all good book stores! This is just a couple of quick hints for people in a hurry.

RPM packages are built from a *spec* file. This consists (in a similar fashion to the KickStart config file) of a recipe of steps that need to be taken in order to build the package – it's expected that you'll have to build it from source, potentially for multiple platforms, and may need to apply patches before compiling. Once built and installed, a binary RPM will be created from the files and directories you specify as being associated with the package. It's important to note that RPM has no idea of which files and directories are related to a given package – you have to tell it.

Here's a sample specification for a custom RPM of the Squid WWW cache server:

```
Summary: Squid Web Cache server
Name: squid
Version: 1.NOVM.22
Release: 1
Copyright: GPL/Harvest
Group: Networking/Daemons
Source: squid-1.NOVM.22-src.tar.gz
Patch: retry-1.NOVM.20.patch
%description
This is just a first attempt to package up the Squid Web Cache for easy
installation on our RedHat Linux servers

%prep
%setup
%build
configure --prefix=/usr/squid
perl -spi -e 's!#( -DALLOW_HOSTNAME_UNDERSCORES)!$1!' src/Makefile
make

%install
make install

%files
/usr/squid
```

Here's how to build this RPM:

```
% mkdir -p SOURCES BUILD SRPMS RPMS/i386
% cp ~/squid-1.NOVM.22-src.tar.gz SOURCES
% cp ~/retry-1.NOVM.20.patch SOURCES
% rpm -ba squid-1.NOVM.22+retry-1.spec
```

This will automatically create a subdirectory under the *BUILD* directory, into which it'll unpack the source code and then apply the patch (there are a number of options available for patching – check the book for details). Now, RPM will automatically build the package by running `configure` and then `make`, install it using `make install`, and take a snapshot of the files under */usr/squid*. It's the latter which will form the binary RPM of the Squid software.

Note that we can insert arbitrary shell commands into the unpacking, building and installing processes, e.g. the call to `perl` which tweaks one of Squid's compile−time parameters.

The final binary RPM will be left under the *RPMS* directory in the platform specific subdirectory *i386*. In this case it will be called *squid−1.NOVM.22−1.i386.rpm*. Note that the filename is created by concatenating the values of the following parameters from the spec file: `Name`, `Version` and `Release` − plus the hardware platform in question, *i386* in this case. Try to bear this in mind when creating your own RPMs, to avoid giving them overly long or painful names!

It's also worth bearing in mind that you can build RPMs without having to rebuild the whole software package, e.g.

```
Summary: Linux 2.0.36 kernel + filehandle patch + serial console patch
Name: linux
Version: 2.0.36+filehandle+serial_console
Release: 1
Copyright: GPL
Group: Base/Kernel
Source: linux−2.0.36+filehandle+serial_console.tar.gz
%description
This is just a first attempt to package up the Linux kernel with patches
for installation on our RedHat Linux servers

%prep
echo

%setup
echo

%build
echo

%install
echo

%post
/sbin/lilo

%files
/lib/modules/2.0.36
/boot/vmlinuz
```

In this case we simply create an RPM based on the */boot/vmlinuz* file and the contents of the directory */lib/modules/2.0.36*, and execute */sbin/lilo* after the package has been installed on a target machine. Let me know if you know much neater way of writing the spec file than this.

14.Appendix C – Munging your own RPMs into the distribution

Here is Eric's script for munging updated RPMs into the RedHat distribution area:

```
#!/usr/bin/perl
#
$redhatdir="/cdrom/i386";
$rpmdir="/cdrom/i386/RedHat/RPMS/";
$updatedir="/cdrom/updates/";
@OTHERDIR=($updatedir);
foreach $dir (@OTHERDIR)
        {
        print "update for $dir\n";
        system(" find $dir -name \"*.rpm\" -exec cp {} $rpmdir \\; ");
        }
chdir($contribdir) || die "peux pas aller dans $contribdir $!\n";
system("chmod -R 755 $redhatdir");
chdir($rpmdir) || die "problem to go in $rpmdir $!\n";
#
# remove the old file
#
opendir(DIR,'.');
@package=grep(/\.rpm$/,readdir(DIR));
foreach $file (@package)
        {
        $file =~ /(.*)\-([\d+|\.]+\w*)\-(\d+)\.[i386|noarch].*/;
        $nom=$1;
        $version=$2;
        $buildvers=$3;
        if ($NOM{$nom})
                {
                $version2=$VERSION{$nom};
                $buildver2=$BUILDVERS{$nom};
                $file2=$FILE{$nom};
                $nom2=$NOM{$nom};
                if ( $version2 gt $version )
                        {
                        print "$file2 is newer than $file\n";
                        unlink($file);
                        }
                else
                        {
                        if ( $version2 lt $version )
                                {
                                print "$file is newer than $file2\n";
                                unlink($file2);
                                $VERSION{$nom}=$version;
                                $BUILDVERS{$nom}=$buildvers;
                                $FILE{$nom}=$file;
                                $NOM{$nom}=$nom;
                                }
                        else
                                {
                                if ( $buildver2 > $buildvers )
                                                {
```

```
                                       print "$file2 : $buildver2 est mieux que $file : $buildvers\n";
                                       unlink($file);
                                            }
                              else
                                            {
                                    print "$file2 : $buildver2 is older than $file : $buildvers\n";
                                        unlink($file2);
                                        $VERSION{$nom}=$version;
                                        $BUILDVERS{$nom}=$buildvers;
                                        $FILE{$nom}=$file;
                                        $NOM{$nom}=$nom;
                                            }
                              }
                      }
              }
      else
              {
              $VERSION{$nom}=$version;
              $BUILDVERS{$nom}=$buildvers;
              $FILE{$nom}=$file;
              $NOM{$nom}=$nom;
              }
      }

# we do the hard thing here
#
system("$redhatdir/misc/src/install/genhdlist $redhatdir");
```