



5

Kernel, cron, and User Administration

CERTIFICATION OBJECTIVES

- | | | | |
|------|-------------------------------------|------|-------------------------|
| 5.01 | Shell Configuration Files | 5.06 | Recompiling a Kernel |
| 5.02 | Setting Up and Managing Disk Quotas | 5.07 | The cron and at Systems |
| 5.03 | The Basics of the Kernel | ✓ | Two-Minute Drill |
| 5.04 | New Kernels, the Easy Way | Q&A | Self Test |
| 5.05 | Kernel Sources | | |

In this chapter, you will learn how to set up the Linux startup shell configuration scripts so that users' sessions are configured according to your (and their) requirements. You will learn how to create and implement policies for managing disk usage—by user or by group.

Next, you will learn how to upgrade, configure, compile, and install your own custom kernels. You will learn about the advantages and disadvantages of monolithic and modular kernels. You have three different ways to customize and optimize your kernel configuration for size and functionality. You will also learn the recommended techniques for configuring and installing the kernel. Finally, you'll learn how to schedule the one-time and periodic execution of jobs.

For the Red Hat exams, the skills you learn in this chapter are important for the Installation and Configuration exam. As described in the Red Hat Exam Prep guide, you need to know how to manage accounts and set up the user environment.

INSIDE THE EXAM

Configuring Users

You need to know how to create and configure users on the Red Hat exams. This means that you need to know how to configure the environment associated with each user account—in configuration files and in user settings. You also need to know how to

specify the configuration files associated with the default bash shell. Finally, you need to know how to limit the resources allocated to each user through quotas. These requirements are all explicitly cited in the Red Hat Exam Prep guide for both exams.

Managing Kernels

As a competent Linux administrator, you need to know how to install, patch, and recompile kernels. It's easy to install a new kernel from an RPM, which makes it a reasonable requirement on the RHCT and RHCE exams. However, the latest RHCE

Exam Prep guide no longer explicitly requires that you need to know how to recompile the Linux kernel. Therefore, the long discussion on recompiling kernels is “only” important if you want the skills needed to become a Linux systems administrator.

CERTIFICATION OBJECTIVE 5.01

Shell Configuration Files

All system-wide shell configuration files are kept in the `/etc` directory. These files are `bashrc`, `profile`, and the scripts in the `/etc/profile.d` directory. These files and scripts are supplemented by hidden files in each user's home directory, as described in Chapter 4. Let's take a look at these files.

`/etc/bashrc`

The `/etc/bashrc` file is used for aliases and functions, on a system-wide basis. Open this file in the text editor of your choice. Read each line in this file. Even if you don't understand the programming commands, you can see that this file sets the following bash shell parameters for each user. For example:

- It assigns a value of `umask`, which creates the default permissions for newly created files. It supports one set of permissions for root and system users (with user IDs below 100), and another for regular users.
- It assigns a prompt, which is what you see just before the cursor at the command prompt.

The settings here are called by the `.bashrc` file in each user's home directory. The settings are supplemented by the `.bash_history` and `.bash_logout` files in each user's home directory.

`/etc/profile`

The `/etc/profile` file is used for system-wide environments and startup files. The following is the profile script from my copy of the RHEL 3 operating system. The first part of the file sets the `PATH` for searching for commands. Then it sets the `PATH`, `USER`, `LOGNAME`, `MAIL`, `HOSTNAME`, `HISTSIZE`, and `INPUTRC` variables, and finally it runs the scripts in the `/etc/profile.d` directory. You can check the current value of any of these variables with the `echo $variable` command.

```
# /etc/profile
```

```
# System wide environment and startup programs, for login setup
```

4 Chapter 5: Kernel, cron, and User Administration

```
# Functions and aliases go in /etc/bashrc

pathmunge () {
    if ! echo $PATH | /bin/egrep -q "^(^|:)$1($|:)" ; then
        if [ "$2" = "after" ] ; then
            PATH=$PATH:$1
        else
            PATH=$1:$PATH
        fi
    fi
}

# Path manipulation
if [ `id -u` = 0 ]; then
    pathmunge /sbin
    pathmunge /usr/sbin
    pathmunge /usr/local/sbin
fi

pathmunge /usr/X11R6/bin after

unset pathmunge

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

USER=`id -un`
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        . $i
    fi
done

unset i
```

/etc/profile.d/

Actually, `/etc/profile.d` is not a script, but a directory of scripts. As I just noted, `/etc/profile` runs the scripts in this directory. Here is a partial listing of the files, which apply to the default bash shell:

```
-rwxr-xr-x  1 root  root    724 Aug 12 11:34 colorls.sh
-rwxr-xr-x  1 root  root    190 Sep  8 11:32 glib2.sh
-rwxr-xr-x  1 root  root     70 Sep 17 12:13 gnome-ssh-askpass.sh
-rwxr-xr-x  1 root  root    210 Sep 23 15:42 krb5.sh
-rwxr-xr-x  1 root  root     53 Mar 26  2003 lam.sh
-rwxr-xr-x  1 root  root   2595 Sep 26 00:39 lang.sh
-rwxr-xr-x  1 root  root    435 Sep  1 10:32 less.sh
-rwxr-xr-x  1 root  root     70 May  1  2003 pvm.sh
-rwxr-xr-x  1 root  root    181 Sep  1 11:01 vim.sh
-rwxr-xr-x  1 root  root    170 Jul 17 15:09 which-2.sh
```

By looking at the `/etc/profile` script, you can see that any script in this directory that ends with an “sh” and is set as an executable will be run when `/etc/profile` is executed.

EXERCISE 5-1

Securing Your System

We want to keep our system as secure as possible. One approach is to change the default permissions users have for new files and directories they make. We'll set all new files and directories to No Access to group or other members.

1. Back up your current `/etc/bashrc` file. If you want to cancel any changes that you make during this exercise, restore from the backup after the final step.
2. Edit the `/etc/bashrc` file. Two lines in the file set the `umask`. One of the two lines is selected depending on the `if` statement above them. See if you can determine which line gets executed for an average (non-root) user.
3. The `if` statement tests to see if the user ID (`uid`) and group ID (`gid`) are the same, and that the `uid` is greater than 99. If this is true, then the first `umask` is executed; otherwise, the second is executed. The second `umask` is for root and other key system accounts. The first is for users.
4. Change the first `umask` statement to exclude all permissions for groups and others. Use `umask 077` to do the job.
5. Save and exit the file.

6. Log in as a nonprivileged user. Use the **touch** command to make a new empty file. Use **ls -l** to verify the permissions on that file.
7. Log in as root. Again, use the **touch** command to make a new empty file and use **ls -l** to verify the permissions on that new file.

You have just changed the default umask for all shell users. If you backed up your `/etc/bashrc` in step 1, you can now restore the original version of this file.

User Shell Configuration Files

As described in Chapter 4, each user gets a copy of the hidden files from the `/etc/skel` directory. As your users start working with their accounts, more configuration files are added to their home directories. Some are based on shells such as bash (`.bash*`); others draw their settings from the GUI desktops that you use, typically GNOME and KDE. I'll describe the GUIs in more detail in Chapter 6.

The default Linux shell is bash, and it's the only shell described in the Red Hat Exam Prep guide and associated courses. However, if you or your users work with other shells, you'll find configuration files associated with those shells hidden in each user's home directory.

CERTIFICATION OBJECTIVE 5.02

Setting Up and Managing Disk Quotas

Quotas are used to limit a user's or a group of users' ability to consume disk space. This prevents a small group of users from monopolizing disk capacity and potentially interfering with other users or the entire system. Disk quotas are commonly used by ISPs, by Web hosting companies, on FTP sites, and on corporate file servers to ensure continued availability of their systems.

Without quotas, one or more users can upload files on an FTP server to the point of filling a filesystem. Once the affected partition is full, other users are effectively denied upload access to the disk. This is also a reason to mount different filesystem directories on different partitions. For example, if you only had partitions for your

root (/) directory and swap space, someone uploading to your computer could fill up all of the space in your root directory (/). Without at least a little free space in the root directory (/), your system could become unstable or even crash.

You have two ways to set quotas for users. You can limit users by inodes or by kilobyte-sized disk blocks. Every Linux file requires an inode. Therefore, you can limit users by the number of files or by absolute space. You can set up different quotas for different filesystems. For example, you can set different quotas for users on the /home and /tmp directories if they are mounted on their own partitions.

Limits on disk blocks restrict the amount of disk space available to a user on your system. Older versions of Red Hat Linux included LinuxConf, which included a graphical tool to configure quotas. As of this writing, Red Hat no longer has a graphical quota configuration tool. Today, you can configure quotas on RHEL only through the command line interface.



Learn to focus on command line tools. Red Hat used to make LinuxConf available as a graphical and console tool for a number of system administration functions, including quotas. While Red Hat may eventually create another GUI quota manager, don't count on it. And GUI tools have been known to crash. On the job, as well as on the exam, command line tools are the only sure way to address just about any Linux configuration issue. Besides, command line tools are faster, and time is often of the essence on the Red Hat exams.

Quota Settings in the Kernel

By default, the Linux kernel as configured by Red Hat supports quotas. However, if you install and compile a new kernel from a remote source, you should make sure that this feature is active. The basic kernel configuration is stored in the /boot directory. For the default RHEL 3 system, you'll find the configuration in the config-2.4.21-4.EL file. If you've configured a custom kernel file, you'll find it listed under a different name.

To verify that quotas are enabled in the default RHEL 3 kernel, run the following command:

```
# grep CONFIG_QUOTA /boot/config-2.4.21-4.EL
```

There are three possible results. If you see the following, quota support is enabled:

```
CONFIG_QUOTA=y
```

Alternatively, if you see the following, quota support is not enabled:

```
CONFIG_QUOTA=n
```

If you don't see any output, then you haven't installed the kernel source files.

If you have a custom or upgraded kernel, use either the **make menuconfig** or **make xconfig** command to make sure support is enabled for quotas. The quota support option is located in the filesystem section. All you need to do is turn on quota support and then rebuild and install your new kernel. I'll describe this process in more detail later in this chapter.

The Quota Package

The quota RPM package is installed by default on RHEL 3, as well as Red Hat Linux 9. you can find out more about RPMs such as quota with the following command:

```
# rpm -qi quota
```

Assuming you haven't removed the quota RPM, you'll see the following description of the package which tells you that it includes a number of tools:

```
The quota package contains system administration tools for
monitoring and limiting user and or group disk usage per
filesystem.
```

You can find out more about these tools, by reviewing a list of associated files. You can find a list of files installed through the quota RPM with the following command:

```
# rpm -ql quota
```

As you can see for yourself, the quota package includes the following commands:

- **/sbin/quotaon /fs** Enables quotas for the /fs filesystem.
- **/sbin/quotaoff /fs** Disables quota tracking.
- **/usr/sbin/edquota name** Edits the quota settings for user *name*. Can also be used to set defaults, or to copy quota settings from one user to another.
- **/usr/bin/quota** Allows users to see their current resource consumption and limits.
- **/usr/sbin/repquota** Generates a report of disk consumption by all users for a quota-enabled filesystem.

- **/sbin/quotacheck** Scans a filesystem for quota usage. Initializes the quota databases.

I've included the entire path to each command for your reference. But as discussed earlier in this book, I recommend that you normally work as the root user during the Red Hat exams. As the noted directories are all part of the root user's PATH, you don't need to specify the full path to each command. (You can verify the directories in your path with the **echo \$PATH** command.)

The next step is to ensure the quotas are active and checked when Linux boots on your system.

sysinit Quota Handling

The `/etc/rc.sysinit` script as described in Chapter 4 initializes Linux system services during the boot process. This script includes commands which start quota services. Specifically, this script runs both the **quotacheck** (to ensure that disk consumption usage records are accurate) and **quotaon** commands (to enable quotas on all filesystems indicated in `/etc/fstab`). You don't have to run these commands manually.

Quota Activation in `/etc/fstab`

As described in Chapter 4, the file `/etc/fstab` tells Linux which filesystems to mount during the boot process. The options column of this file configures how Linux mounts a directory. You can include quota settings in `/etc/fstab` for users and or groups.



Before you edit a key configuration file such as `/etc/fstab`, it's a good idea to back it up and save it to any boot or rescue disks that you may have. If your changes lead to a catastrophic failure, you can boot your system from a rescue disk and then restore the original configuration file.

Here is a sample `/etc/fstab` before editing:

Device	Mount point	Filesys	Options	dump	Fsck
LABEL=/	/	ext3	defaults	1	1
LABEL=/boot	/boot	ext3	defaults	1	2
none	/dev/pts	devpts	gid=5,mode=620	0	0
none	/proc	proc	defaults	0	0
none	/dev/shm	proc	tmpfs	0	0
/dev/hda3	swap	swap	defaults	0	0
/dev/hdd1	/home	ext3	defaults	1	2
/dev/cdrom	/mnt/cdrom	udf,iso9660	noauto,owner,kudzu,ro	0	0
/dev/floppy	/mnt/floppy	auto	noauto,owner,kudzu	0	0

In this configuration, we may want to enable quotas on the root (/) and /home directory filesystems. You can tell Linux to start tracking user quotas by adding the keyword **usrquota** under the options column. Similarly, you can tell Linux to start tracking group quotas with the **grpquota** option. Use vi or your favorite text editor to update /etc/fstab.

In our example, we will add both user and group quotas to the /home directory filesystem:

Device	Mount point	Filesys	Options	dump	Fsck
LABEL=/	/	ext3	defaults	1	1
LABEL=/boot	/boot	ext3	defaults	1	2
none	/dev/pts	devpts	gid=5,mode=620	0	0
none	/proc	proc	defaults	0	0
none	/dev/shm	proc	tmpfs	0	0
/dev/hda3	swap	swap	defaults	0	0
/dev/hdd1	/home	ext3	exec,dev,suid,rw,usrquota,grpquota	1	2
/dev/cdrom	/mnt/cdrom	udf,iso9660	noauto,owner,kudzu,ro	0	0
/dev/floppy	/mnt/floppy	auto	noauto,owner,kudzu	0	0

If you edit the /etc/fstab file by hand, you'll need to ensure that the line you are editing does not wrap to the next line. If it does, the format for your /etc/fstab will be invalid and you may not be able to successfully boot Linux.



You can test changes to /etc/fstab by rebooting your computer or remounting a filesystem. For example, if you've just added *usrquota* and *grpquota* entries as shown to the */home* directory filesystem, you can test it with the *mount -o remount /home* command.

Quota Management Commands

The next step is to create quota files. For user and group quotas, you'll need the aquota.user and aquota.group files in the selected filesystem. You need these files before you can activate actual quotas. You no longer need to create those files; once you've remounted the desired directory, you can create them directly with the appropriate **quotacheck** command. For the /home directory described earlier, you'd use the following commands:

```
# mount -o remount /home
# quotacheck -avugm
```

The **quotacheck -avugm** command automatically scans `/etc/mtab`, which includes the mounted directories from `/etc/fstab`. The options for **quotacheck** are:

- **-a** Scans all filesystems with quotas enabled by checking `/etc/mtab`.
- **-v** Performs a verbose scan.
- **-u** Scans for user quotas.
- **-g** Scans for group quotas.
- **-m** Remounts the scanned filesystem.

This will check the current quota information for all users, groups, and partitions. It stores this information in the appropriate quota partitions. You should also find the `aquota.user` and `aquota.group` files in the configured directory. If you're configuring quotas on the `/home` directory, you can check that it worked with the following command:

```
# ls -l /home/aquota.*
```

Using edquota to Set Up Disk Quotas

To specify disk quotas, you need to run the **edquota** command. This edits the `aquota.user` or `aquota.group` file with the `vi` editor. In our example, we will pretend we have a user named `nancy`, and we want to restrict how much disk space she is allowed to use. We type the following command to edit their quota records:

```
# edquota -u nancy
```

This command launches the `vi` editor and opens the quota information for user `nancy` as shown in Figure 5-1.

The quota information is formatted strangely. There are actually seven columns. The lines are wrapped. In this case, the filesystem with the quota is mounted on partition `/dev/hdd1`. There are soft and hard limits for both inodes and files. By default, soft and hard limits of 0 means that there are no limits for `nancy`.

FIGURE 5-1

Quota
information

Disk quotas for user nancy (uid 507):						
Filesystem	blocks	soft	hard	inodes	soft	hard
hard						
/dev/hdd1	52	0	0	13	0	0
	0					

We can see that nancy is currently using 52 blocks and has 13 files (inodes) on this partition. Each block takes up 1KB of space; thus user nancy’s files total 52KB. We want to set a limit so that nancy does not take more than 20MB of space with her files.

First, we need to elaborate on the meaning of soft and hard limits.

- **Soft limit** This is the maximum amount of space a user can have on that partition. If you have set a grace period, then this will act as an alarm. The user will then be notified he is in quota violation. If you have set a grace period, you will also need to set a hard limit. A grace period is the number of days a user is allowed to be above the given quota. After the grace period is over, the user must get under the soft limit to continue.
- **Hard limit** Hard limits are necessary only when you are using grace periods. If grace periods are enabled, this will be the absolute limit a person can use. Any attempt to consume resources beyond this limit will be denied. If you are not using grace periods, the soft limit is the maximum amount of available to each user.

In our example, we will set our user an 18MB soft limit and a 20MB hard limit. As shown in Figure 5-2, this is written as a number of 1KB blocks in the quota file.

Note that we have not limited user nancy’s use of inodes. She is still able to use as many inodes (thus as many files) as she likes. To implement these quotas, we must save these settings. Assuming you’re still using the default vi editor, the `:wq` command does this job nicely.

We will also give user nancy a seven-day grace period, if and when she exceeds the soft limit. She has that amount of time to get back under the soft limit. To set the grace period for all users, run the `edquota -t` command. The result should look similar to what you see in Figure 5-3:

Here, Linux has provided us with the default of seven days for both inodes and block usage. That is, a user may exceed his soft limit on either resource for up to seven days. After that, further requests by that user to use files will be denied. Our

FIGURE 5-2

Quotas with hard and soft limits

Disk quotas for user nancy (uid 507):					
Filesystem	blocks	soft	hard	inodes	soft
hard					
/dev/hdd1	52	18000	20000	13	0
0					

FIGURE 5-3

Quota grace
period

Grace period before enforcing soft limits for users:					
Time units may be: days, hours, minutes, or seconds					
Filesystem	Block grace period		Inode grace period		
/dev/hdd1	7days		7days		

user nancy would have to delete files to get her total disk block consumption under 18MB before she could create new files or grow existing files. You can edit the grace period directly, using vi commands. To activate the new grace period, just save the file.

There is a quirk to quota grace periods. When you use **edquota** and specify the grace period, you cannot have a space between the number and the unit (for example, **7days**, not **7 days**). Fortunately, the quota system in RHEL 3 automatically fixes this problem.



In older versions of Red Hat Linux, a space between the number and the unit would lead to a quota error.

The **edquota** command allows you to use an already configured user's quota as a template for new users. To use this feature, you need to add the following switch and options, **-p configured_user** arguments:

```
# edquota -up nancy michael randy donna
```

This command will not provide any output, but it will take the quota configuration settings of user nancy and apply them to michael, randy, and donna. You can list as many users as you want to edit or apply templates to.

You can also set up quotas on a per-group basis. To do this, simply run **edquota** with the **-g group_name** argument. Here, *group_name* would need to be a valid group as specified in the */etc/group* file.

```
# edquota -g nancy
```

This opens the block and inode quota for group nancy, as shown in Figure 5-4.

FIGURE 5-4

Group quota

Disk quotas for group nancy (gid 507):					
Filesystem	blocks	soft	hard	inodes	soft
hard					
/dev/hdd1	52	0	0	13	0
0					

Automating Quota Settings

As an administrator, you'll want to maintain any quotas that you create. For that purpose, it's useful to run the aforementioned **quotacheck** command on a regular basis. As you'll see later in this chapter, that is easy to do through the cron system. A simple command in the right cron file automatically runs the **quotacheck** command on a regular basis. For example, the following command in the right cron file runs the **quotacheck** command at 4:00 A.M. every Saturday:

```
0 4 * * 6 /sbin/quotacheck -avug
```

You can also use the **edquota** command to apply quotas to all users on your system. For example, the following command applies the quotas that you've already set on user **mj** to all other real users on the system:

```
edquota -p mj `awk -F: '$3 > 499 {print $1}' /etc/passwd`
```

Note that this command lists the first column (\$1) of **/etc/passwd**, which is the user name. And in keeping with the UIDs for regular Red Hat users (from the third column, \$3, of **/etc/passwd**), this is limited to users with UIDs of 500 or higher. You can add this type of command to the appropriate cron file as well, which makes sure that the quotas are applied to all existing and new users.

Quota Reports

As an administrator, it can be useful to see reports on who is using the most disk space. You can generate reports on users, groups, or everybody on every partition. To view a report showing quota information for all, run the **repquota -a** command. You'll see a list of quotas for all users similar to what is shown in Figure 5-5.

If you have multiple filesystems with quotas, you can use the **repquota** command to isolate a specific filesystem. For example, if you wanted to view the quota report for the partition with the **/home** directory, you'd run the following command:

```
# repquota -u /home
```

Alternatively, if you wanted to view quota information on user **nancy**, run the following **quota** command:

```
# quota -uv nancy
Disk quotas for user nancy(uid 507):
Filesystem blocks quota limit grace files quota limit
grace
/dev/hdd1      52  18000  20000          13      0      0
```

An individual user can check his or her own usage with the **quota** command, but only the administrative root user can examine the quotas for other users.

Quotas on NFS Directories

The Network File System (NFS) allows users to share files and directories on a network with Linux and Unix computers. Users across the network mount a shared NFS directory from a specific computer. Users are normally in a single database in an NFS setup. Disk quotas can be applied to these users in virtually the same way as on a regular Linux computer. For example, if you create a local user called **nfsuser**, and you translate all remote requests to this user, then you need to set up quota restrictions for **nfsuser** on the mounted partition. This will limit the disk consumption of all incoming NFS users. See Chapter 9 for more about NFS.

FIGURE 5-5 A quota report

```
[root@Enterprise3 root]# repquota -a
*** Report for user quotas on device /dev/hdd1
Block grace time: 7days; Inode grace time: 7days

```

User	Block limits				File limits			
	used	soft	hard	grace	used	soft	hard	grace
root	-- 36	0	0		3	0	0	
michael	-- 52	18000	20000		13	0	0	
donna	-- 52	0	0		13	0	0	
elizabeth	-- 52	0	0		13	0	0	
nancy	-- 52	18000	20000		13	0	0	
randy	-- 52	18000	20000		13	0	0	

```
[root@Enterprise3 root]#
```

EXERCISE 5-2

Configure Quotas

In this exercise, we will set up user quotas for one user on your system. These quotas will allow a soft limit of 80MB and a hard limit of 100MB for each user.

No limits are to be placed on the number of inodes. Assume the /home directory is mounted on a separate partition. (If /home is not mounted separately, apply the commands to the top-level root directory /.) The first couple of steps should be formalities, as quotas should be active and installed by default. However, it's a good habit to check. To set up quotas in this exercise, use the following steps:

1. Check your kernel configuration for the CONFIG_QUOTA variable, using the /boot/config-2.4.21-4.EL file. It should be set to "Y." If not, proceed to the Lab Question at the end of this chapter for instructions on how to revise your kernel. If you're using a different version of Linux such as Red Hat Linux 9, substitute the /boot/config-* file associated with your kernel version.
 2. Check to make sure that the quota package is installed. Install from the RHEL 3 installation source if required.
 3. Add quotas to /etc/fstab. Add the usrquota variable to the Options column for the partition with the /home directory. Make sure the info stays on one line in /etc/fstab.
 4. Activate the quotas. You can unmount and remount the /home directory, reboot Linux, or use the following command:

```
# mount -o remount /home
```
 5. Use the **quotacheck -avum** command to activate the quota files in the /home directory.
 6. Make sure this command worked. Look for the aquota.user file in the /home directory.
 7. Now you're ready to set up quotas for a specific user. If necessary, look up usernames in /etc/passwd. Use the **edquota -u username** command to edit the quotas for the user of your choice.
 8. Under the soft and hard columns, change the 0 to 80000 and 100000, respectively. Remember, these files are set up for 1KB blocks. Save the file.
-

CERTIFICATION OBJECTIVE 5.03

The Basics of the Kernel

The kernel is the heart of the whole operating system. It manages communication with hardware, decides which processes to run, and provides each process with an isolated, virtual address space in which to run. The kernel is what your boot loader, GRUB or LILO, loads into memory. The kernel loads device driver modules. It also allocates hardware resources such as IRQ ports, I/O addresses, and DMA channels. When you recompile your kernel, you can

- Greatly improve the speed at which kernel services operate.
- Build in direct support for commonly used drivers.
- Configure the dynamic loading of appropriate drivers as modules.
- Lower the memory consumption of your kernel by removing unneeded components.
- Configure support for high-end hardware, such as memory above 4GB, hardware array controllers, symmetric multiprocessing (multiple CPU) support, and more.

In essence, you can customize the Linux kernel any way you want. The best way to do it is to make it fit every detail of your hardware. However, you may not need to be so picky. In many cases, where there's a small update to the kernel, all you need to do is install the updated kernel RPM.

Best Practices

You should compile your kernel with only the elements you need. The more you can leave out, the faster your whole system will run. For example, if you don't have a sound card, you can remove sound card support from your kernel. By removing unneeded devices, you will

- Decrease the size of the kernel.
- Provide a modest increase in speed for the devices that are present.

- Make more hardware resources (I/O addresses, IRQ ports, and so on) available for other hardware such as network cards, disk controllers, and more.
- Reduce the chance of hardware limits, such as those that may be based on the size of the compressed kernel.

Generally, it is a good idea to have device drivers compiled as modules for any equipment that you may add in the near future. For example, if you may use your Linux computer as a router, you'll need a second network card, and you can add support for that card to your kernel. For example, if you have a 3Com 3c595 network card installed but you also have some 3Com 3c905 cards in storage, then it may be a good idea to include the 3c905 module. That way, you will just have to swap in the new card and let the module load, causing minimum downtime.

Modules are kernel extensions. They are not compiled directly into the kernel but can be plugged in and removed as needed. When configured as a module, a hardware failure such as that of a network card will not cause the whole system to fail.

Kernel Concepts

You will need to understand some basic kernel concepts before you can compile your own kernel. Kernels can be organized as one big unit, or as a lot of interconnected pieces. Kernels are called up by boot loaders when you boot your computer.

Monolithic versus Modular

A *monolithic* kernel is a kernel where all the device modules are built directly into the kernel. *Modular* kernels have many of their devices built as separate loadable modules. Monolithic kernels can communicate with devices faster, since modular kernels can talk to the hardware only indirectly through a module table. Unfortunately, monolithic Linux kernels are huge. Bigger kernels reduce available RAM. In addition, some systems just can't boot a kernel that's too large.

There used to be advantages to a monolithic kernel. Linux once had problems loading modular kernels for some hardware. With a monolithic kernel, the drivers would already be there. But now modular kernels load new drivers a lot more reliably.

A modular kernel has greater flexibility. You can compile almost all your drivers as modules, and then each module can be inserted into the kernel whenever you need it. Modules keep the initial kernel size low, which decreases the boot time and improves overall performance. If Linux has trouble loading a kernel module, you can use the **modprobe** or **insmod** commands to load modules as needed.

Updating the Kernel

Updating the kernel is not as difficult as it looks. You should always keep a copy of your old kernel around in case you make a mistake. New kernels are handled by installing the newly built kernel in /boot and then adding another boot option to your boot loader configuration file (/etc/grub.conf or /etc/lilo.conf) for the new kernel. GRUB or LILO treats the new kernel as if it were an entirely new operating system.

If you install the new kernel directly from a Red Hat configured RPM, it updates your boot loader automatically.

If you do make a drastic mistake and the kernel doesn't boot, then you can simply reboot the server and select your old kernel at the GRUB or LILO prompt. You should also save your kernel configuration files so that you can easily copy to the newer kernels and use them as a guideline. This will be discussed in more detail later in this chapter.

Other RHEL 3 Kernels

There are a number of different kernels included with the RHEL 3 installation files. You can and should install the kernel best suited to your system. I briefly describe available RHEL 3 kernels in Table 5-1. The "EL" in each of these kernels refers to their customization for Red Hat Enterprise Linux. The version numbers shown is what was released with RHEL 3. If you've used the Red Hat update agent, your kernel version number may vary.

TABLE 5-1 Available Red Hat Enterprise Linux 3 Kernels (and Related Packages)

Kernel RPM	Description
kernel-2.4.21-4.EL.athlon.rpm	Suitable for PCs with a single AMD Athlon CPU.
kernel-2.4.21-4.EL.i686.rpm	Designed for PCs with a single Intel CPU.
kernel-unsupported-2.4.21-4.EL.athlon.rpm	The Athlon kernel with additional unsupported modules.
kernel-unsupported-2.4.21-4.EL.i686.rpm	The Intel kernel with additional unsupported modules.
kernel-BOOT-2.4.21-4.EL.i386.rpm	Kernel used only during the RHEL 3 installation process.
kernel-hugemem-2.4.21-4.EL.i686.rpm	Supports multiple CPUs and systems with more than 4GB of RAM.

TABLE 5-1 Available Red Hat Enterprise Linux 3 Kernels (and Related Packages) (continued)

Kernel RPM	Description
kernel-hugemem-unsupported-2.4.21-4.EL.i686.rpm	The <i>hugemem</i> kernel configured with additional untested kernel modules.
kernel-pcmcia-cs-3.1.31-13.i386.rpm	Adds PCMCIA (PC Card) modules to your current kernel.
kernel-smp-2.4.21-4.EL.athlon.rpm	The symmetric multiprocessing (SMP) kernel suitable for multi-CPU AMD Athlon systems. Also supports more than 4GB of RAM.
kernel-smp-2.4.21-4.EL.i386.rpm	The symmetric multiprocessing (SMP) kernel suitable for multi-CPU Intel systems. Also supports more than 4GB of RAM.
kernel-smp-unsupported-2.4.21-4.EL.athlon.rpm	The SMP kernel with additional untested kernel modules for Athlon CPUs.
kernel-smp-unsupported-2.4.21-4.EL.i386.rpm	The SMP kernel with additional untested kernel modules for Intel CPUs.
kernel-source-2.4.21-4.EL.i386.rpm	Includes the source code for the RHEL 3 kernel.

This is just a short list of kernels available for RHEL 3. As the Red Hat exams assume the use of standard PCs with a single CPU, I’ve limited the list in Table 5-1 to such kernels. For more information on RHEL 3 kernels available for multi-CPU or higher-end CPUs, refer to the RHEL 3 documentation available online from www.redhat.com/docs/manuals/enterprise/.

The /boot Partition

The Linux kernel is stored in the partition with the /boot directory. New kernels must also be transferred to this directory. By default, RHEL 3 configures a partition of about 100MB for this directory. This provides enough room for your current kernel plus several additional upgraded kernels.

The /proc Filesystem

The /proc directory is based on a virtual filesystem; in other words, it does not include any files that are stored on the hard drive. But it is a window into what the kernel sees of your computer. It’s a good idea to study the files and directories in /proc, as it can

FIGURE 5-6 A Red Hat Enterprise Linux 3 /proc directory

```

[root@Enterprise3 root]# \ls /proc/
1      1880 1950 2159 2245 2762      devices      ksyms        stat
1175   1903 1960 2170 2246 3         dma          loadavg      swaps
1182   1907 1969 2171 2254 4         driver       locks        sys
16     1908 1999 2174 2256 5         execdomains  mdstat       sysrq-trigger
1655   1909 2     2191 2261 6         fb           meminfo      sysvipc
1659   1910 2008 2216 2365 7         filesystems  misc         tty
1685   1911 2022 2218 2420 74        fs           modules      uptime
17     1912 2023 2223 2485 8         ide          mounts       version
1704   1913 2024 2230 2586 9         interrupts  mtrr
1772   1914 2025 2234 2655 apm        iomem        net
18     1915 2026 2236 2713 bus        ioports      partitions
1811   1916 2027 2238 2714 cmdline   irq          pci
1851   1922 2028 2241 2715 cpuinfo    kcore        self
1865   1941 2085 2243 2749 crypto     kmsg         slabinfo
[root@Enterprise3 root]#

```

help you diagnose a wide range of problems. Figure 5-6 shows the /proc from a typical RHEL 3 computer.

The numbered items are based on process IDs. For example, the process ID of `init` is 1. The files in this directory include the memory segments that make up the active process. The contents of each of these files include the active memory for that process.

The other items in the listing are files and directories that correspond to configuration information for components such as DMA channels or whole subsystems such as memory information.

Take a look at some of these files. For example, the `/proc/meminfo` file provides excellent information as to the state of memory on the local computer, as shown in Figure 5-7. It can help you determine if RHEL 3 is having trouble detecting all of the memory on your computer.

It can also help you measure the current memory state of your system. For example, if your system is overloaded, you'll probably find very little free swap space. The HugePage settings are associated with systems with over 4GB of RAM.

Now you can examine how Linux looks at your CPU in the `/proc/cpuinfo` file, as shown in Figure 5-8. In this particular case, the `cpu` family information is important; the number 6 in this figure corresponds to a 686 CPU. Some of this information is available through the `top` utility.

Many programs are available that simply look at the information stored in /proc and interpret it in a more readable format. The `top` utility is a perfect example. It reads the process table, queries RAM and swap usage and the level of CPU use, and presents it all on one screen.

FIGURE 5-7

Detected
memory
information

```
[root@Enterprise3 root]# cat /proc/meminfo
              total:      used:      free:  shared: buffers:  cached:
Mem: 128675840 126464000  2211840          0 14028800 63885312
Swap: 394805248 31461376 363343872
MemTotal:      125660 kB
MemFree:        2160 kB
MemShared:         0 kB
Buffers:        13700 kB
Cached:         51304 kB
SwapCached:     11084 kB
Active:         87704 kB
ActiveAnon:     49664 kB
ActiveCache:    38040 kB
Inact_dirty:        0 kB
Inact_laundry:   18908 kB
Inact_clean:     2208 kB
Inact_target:    21764 kB
HighTotal:         0 kB
HighFree:         0 kB
LowTotal:       125660 kB
LowFree:         2160 kB
SwapTotal:      385552 kB
SwapFree:       354828 kB
HugePages_Total:  0
HugePages_Free:   0
Hugepagesize:    4096 kB
[root@Enterprise3 root]#
```

IP Forwarding

More importantly, there are kernel variables you can alter to change the way the kernel behaves while it's running. Sometimes it's appropriate to configure a Linux computer as a router between networks. By default, it does not forward TCP/IP information. You can confirm it with the following command:

```
# cat /proc/sys/net/ipv4/ip_forward
0
```

If your computer has two or more network cards, you may want to activate *IP forwarding* with the following command:

```
# echo 1 >> /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
```

FIGURE 5-8

```

processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 11
model name    : Intel(R) Pentium(R) III Mobile CPU      1200MHz
stepping      : 1
cpu MHz       : 1193.582
cache size    : 512 KB
fdiv_bug      : no
hlt_bug       : no
f00f_bug      : no
coma_bug      : no
fpu           : yes
fpu_exception : yes
cpuid level   : 2
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat pse
36 mmx fxsr sse
bogomips      : 2365.84
[
~
~
~
~
~
~
~
"/proc/cpuinfo" [readonly] 19L, 402C                                19,0-1      All

```

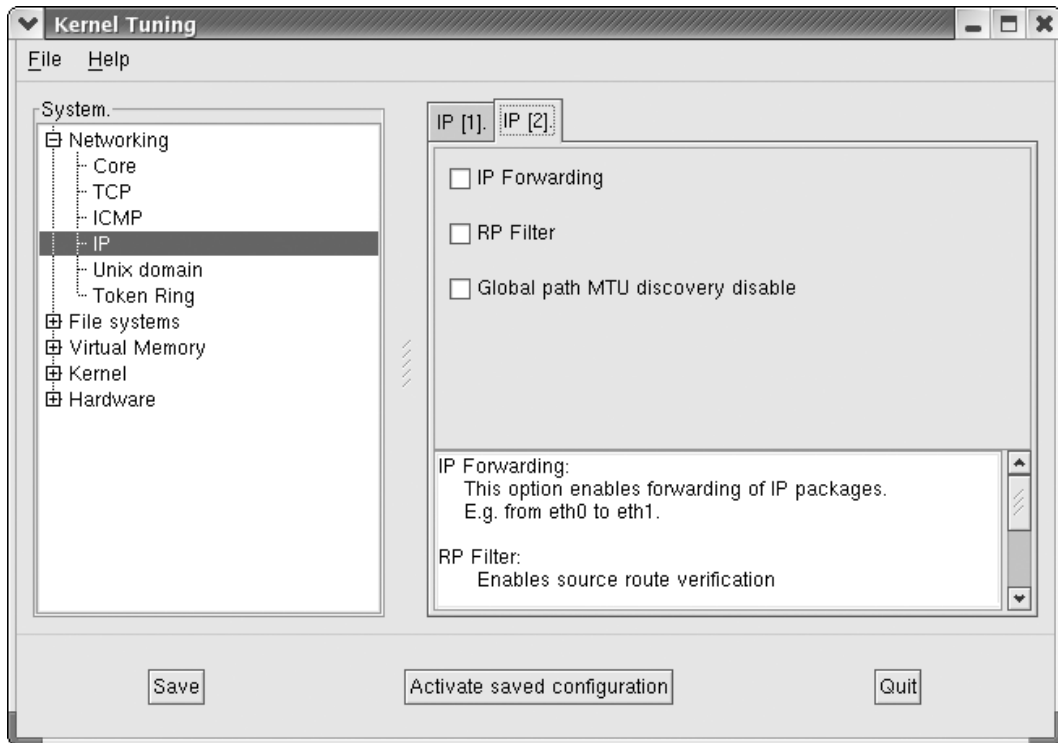
Preventing the Ping of Death

The following is another useful change to a proc kernel variable, which enables the use of TCP SYN packet cookies. These cookies prevent SYN flood attacks on your system, including the so-called “ping of death.”

```
# echo 1 >> /proc/sys/net/ipv4/tcp_syncookies
```

Managing /proc Graphically

There is a Red Hat graphical tool that you can use to manage /proc directories. It’s known as the Kernel Tuning tool, which you can start from a GUI command line with the **redhat-config-proc** command. For example, you can use it to set up IP Forwarding as shown in Figure 5-9.

FIGURE 5-9 Tuning the kernel through /proc

Understanding Kernel Modules

When you compile your kernel, you can set up a monolithic kernel with every driver that you might ever need. Unfortunately, such kernels are large, unwieldy, and take a lot of time to load. Generally, most Linux administrators use kernel modules. As described earlier, a *kernel module* is not compiled directly into the kernel but instead operates as a pluggable driver that can be loaded and unloaded into the kernel as needed.

exam Watch

If you're having problems with hardware related kernel modules, one utility you can try is kudzu, the Red Hat hardware detection and configuration utility. It is normally run during the boot process, but you might try running it again if you're having a problem.

To have the kernel dynamically load and unload kernel modules as needed, the kernel module loader, **kmod**, is used to control the loading and unloading of modules. For special parameters and options, edit the `/etc/modules.conf` file.

Most hardware modules are automatically detected. If you've just installed new undetected hardware, you could issue the following command:

```
# depmod -a
```

This will scan through your modules, find out what the different dependencies for all your modules are, and map them out to a file (`modules.dep`). This command also creates a number of other files in the `/lib/modules/2.4.21-4.EL/` directory (if you've installed another kernel, the directory version changes accordingly).

RHEL 3 runs this command automatically through the `/etc/rc.sysinit` script. Once the **depmod** module scan is complete, you can load additional kernel modules. If that module has dependencies, then all the needed modules will automatically load first.

To load a module, you can use the **modprobe** command with the name of a specific driver:

```
# modprobe 3c503
```

In this example, the Ethernet module for a 3Com 503 network card requires the 8390 module to work properly. If **depmod** was run first, then 8390 would have loaded automatically before the 3c503 driver. If a dependency in the list fails during loading, then all modules will be automatically unloaded.

Alternatively, you can set up these modules in `/etc/modules.conf`. It should already be configured during the RHEL 3 installation process. Unfortunately, this work can be rather tedious. The following commands are accepted in this file:

- **alias** Allows you to bind a name to a module.
- **options** Allows you to specify options for a module.
- **install *module command*** Use *command* instead of **insmod** on this module.
- **pre-install *module command*** Run *command* before installing this module.
- **post-install *module command*** Run *command* after installing this module.
- **remove *module command*** Use *command* instead of **rmmod** on this module.
- **pre-remove *module command*** Run *command* before loading this module.
- **post-remove *module command*** Run *command* after loading this module.

Here is an example of what a common `modules.conf` file may look like:

```
alias eth0 pcnet32
alias usb-controller usb-uhci
options sb irq=5 io=0x220 dma=1
alias midi awe_wave
alias parport_lowlevel parport_pc
```

Here the `eth0` device is bound to the `pcnet32` module. To load the network card, you can then simply type **`modprobe eth0`** without knowing what card is in the computer. The next command sets the USB controller. The following two lines show the configuration of a soundblaster (`sb`) module. This information includes a specific IRQ port, I/O address, and DMA channel. The options line specifies these options and binds them to the `sb` alias. The sound card happens to be a Sound Blaster AWE 32 model card; therefore, the `midi` alias is bound to the `awe_wave` module. Finally, a parallel port module is bound to the `parport_lowlevel` alias.

The `/etc/rc.sysinit` script recognizes certain aliases and will load them if it finds them in this file. You need to specifically place the sound modules in `modules.conf` to have them automatically loaded. To have the sound modules automatically loaded during the Linux boot process without having to edit the `/etc/rc.sysinit` file, you can simply create an alias to sound and or midi in the `modules.conf` file.

To see what modules are loaded, you can type either

```
# cat /proc/modules
```

or

```
# lsmod
```

Both commands return output that looks something like the following:

Module	Size	Used by	Not tainted
nfs	92912	1 (autoclean)	
ide-cd	35680	0 (autoclean)	
cdrom	33696	0 (autoclean)	[ide-cd]
smbfs	44528	1 (autoclean)	
nfsd	85456	8 (autoclean)	
lockd	59856	1 (autoclean)	[nfs nfsd]
sunrpc	85692	1 (autoclean)	[nfs nfsd lockd]
parport_pc	19076	1 (autoclean)	
lp	9028	0 (autoclean)	
parport	37088	1 (autoclean)	[parport_pc lp]
autofs	13364	0 (autoclean)	(unused)
pcnet32	18080	1	

mi	3976	0	[pcnet32]
crc32	3712	0	[pcnet32]
ipt_REJECT	4632	1	(autoclean)
ipt_state	1080	1	(autoclean)
ip_conntrack	27304	1	(autoclean) [ipt_state]
iptables_filter	2412	1	(autoclean)
ip_tables	15776	3	[ipt_REJECT ipt_state iptable_filter]
floppy	58160	0	(autoclean)
microcode	4724	0	(autoclean)
keybdev	2976	0	(unused)
mousedev	5524	1	
hid	22212	0	(unused)
input	5888	0	[keybdev mousedev hid]
usb-uhci	26412	0	(unused)
usbcore	79392	1	[hid usb-uhci]
ext3	91592	3	
jbd	52336	3	[ext3]
raid1	14988	2	

The module name is listed on the left, and its size is in the second column. The “Used by” column shows more detail on how the module is being handled. An autoclean message means that the kernel, using the kmod thread is taking care of the module and will handle removing it. If a module name, such as ext3, is listed in brackets, then the module depends on the module in brackets. In our example, jbd depends on the ext3 module.

The `/lib/modules/kernel_version/` Directory Structure

All kernel modules are stored in the `/lib/modules/kernel_version/` directory. When RHEL 3 is installed, `kernel_version=2.4.21-4.EL`. If you have recently compiled a new kernel and your modules are not loading properly, then you have probably forgotten to compile and install the modules. In the `/usr/src/kernel_version` source directory, run the following commands:

```
# make modules
# make modules_install
```

The first line compiles the modules, while the second places them under the proper directory tree. In this directory tree, different subdirectories represent different groupings. The following is a sample of a module directory:

```
# ls -l /lib/modules/2.4.21-4.EL/kernel/drivers
total 60
drwxr-xr-x  7 root      root          4096 Oct 23 15:22 addon
```

```

drwxr-xr-x    2 root    root          4096 Oct 23 15:22 block
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 cdrom
drwxr-xr-x    8 root    root          4096 Oct 23 15:22 char
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 ide
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 input
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 md
drwxr-xr-x    3 root    root          4096 Oct 23 15:22 message
drwxr-xr-x   10 root    root          4096 Oct 23 15:22 net
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 parport
drwxr-xr-x    2 root    root          4096 Oct 23 15:22 pcmcia
drwxr-xr-x    5 root    root          4096 Oct 23 15:22 scsi
drwxr-xr-x    5 root    root          4096 Oct 23 15:22 sound
drwxr-xr-x    5 root    root          4096 Oct 23 15:22 usb
drwxr-xr-x    3 root    root          4096 Oct 23 15:22 video

```

Remember that each `/lib/modules/kernel_version` directory contains a `modules.dep` file that lists all the dependencies for all the modules within the directories. Each of these module directories includes a group of kernel modules for a common type of hardware. You might want to become familiar with where to find certain modules when needed. Here are some module types you can find under each directory:

- **addon** High-end cards such as for server racks
- **block** Block devices: parallel port ide drives, network block devices, XT disks, hardware raid devices
- **cdrom** Non-ATAPI CD-ROM drivers: Mitsumi, Sony
- **char** Miscellaneous input and serial devices
- **ide** Hard disk drivers
- **input** Input devices (keyboards, mice)
- **md** raid devices
- **message** Specialized I/O adapters
- **net** Network modules: basic network cards, generic ppp, slip
- **parport** Parallel port devices (not printers)
- **pcmcia** Drivers used by the `pcmcia cardmgr` daemon (the actual cards use separate drivers)
- **scsi** SCSI tape, RAID, and hard drive modules, video (special video modules for Linux)
- **sound** Sound adapters

- **usb** Universal Serial Bus hubs and devices
- **video** Graphics adapters

All modules have `.o` for an extension (such as `pcnet32.o`). You do not need to specify the full name, just the first part of the module file (`pcnet32`). Once you know the directory structure, you can have the **modprobe** command load all modules for a certain category. For instance, if you are on a PC and you don't know the network card, you can simply type

```
modprobe -t net
```

This will attempt to load all modules in `/lib/modules/kernel_version/net`, stopping when a match is found. To remove a module such as `pcnet32` and all its dependencies, you can type either

```
modprobe -r pcnet32
```

or

```
rmmod -r pcnet32
```

Either of these commands will remove the modules and all their dependencies, provided they are not in use by another module or not currently active. For example, if your network is active, you can't remove the network `pcnet32` driver module. If you want to remove only the module and leave the other dependent drivers, run the **rmmod** command without the `-r` switch.

CERTIFICATION OBJECTIVE 5.04

New Kernels, the Easy Way

On the Red Hat exams, you may be expected to upgrade your kernel by installing the latest Red Hat kernel RPM. You may be able to patch an existing kernel. Whatever you do, you should make a boot disk associated with your new or upgraded kernel with the **mkbootdisk** command.

Before we begin, it's important to understand the way kernels are numbered.

Understanding Kernel Version Numbers

The version number associated with the RHEL 3 kernel may look a little confusing, but it tells you a lot about the history of the kernel. The standard RHEL 3 kernel is version 2.4.21-4.EL, formatted in a *majorversion.majorrevision.patch-build* format.

The first number (2) is the major version number. These are drastic changes to the kernel. Typically, older version software will *not* work in the newer version when this number changes. Kernel major version numbers are reserved for completely new kernel designs.

The second number (4) actually has two meanings. First, it indicates this is the fourth major revision of major version 2 of the kernel. Second, since it is an even number, it indicates that the kernel release is a stable release. If it were an odd number, it would be a developmental kernel, not suitable for production computers.



While Linux kernel 2.6 has been officially released, Red Hat has already incorporated a number of associated features in the RHEL 3 kernel. To promote stability, Red Hat has no plans as of this writing to incorporate or offer a version of Linux kernel 2.6 for RHEL 3. This is consistent with the demands of the RHEL 3 customer base; most business customers want to stay away from the “bleeding edge.”

The third number (21) is the patch version number for the kernel. These changes are typically small changes, bug fixes, security fixes, and enhancements. Generally, you can use the `zcat` command to increment one patch at a time. For example, if your current kernel is version 2.4.21, you can use the `patch-2.4.22.gz` file to upgrade your kernel to version 2.4.22.

The fourth number (-4.EL) is a number added by Red Hat. This is the fourth Red Hat build of Linux kernel 2.4.21, which incorporates features customized for Red Hat Enterprise Linux.

Some Linux software, such as firewalls, are associated with a specific Linux kernel. The requirements are associated only with the first two major numbers. For example, the older `ipchains` firewall command works only with version 2.2 and later kernels. In contrast, the `iptables` firewall commands require version 2.4 and later kernels.

Upgrading Kernels

During the lifetime of RHEL 3, you may find a security advisory which strongly suggests that you upgrade your Linux kernel. In this case, a Red Hat kernel RPM will be available through the Red Hat Network.

 exam**Watch**

You won't have access to the Internet during the Red Hat exams, and therefore, you may not be able to get to the Red Hat Network for updates. However, you may still be required to install an upgraded kernel.

Upgrading a kernel from a Red Hat RPM is fairly easy. Basically, all you need to do is install the new kernel with the appropriate `rpm` command. When properly configured, the RPM automatically upgrades your default boot loader as well. For example, say you've just downloaded the 2.4.21-4.1.EL kernel RPM from one of the FTP servers with Red Hat RPMs to the `/tmp` directory.

 exam**Watch**

If you're told to upgrade a new kernel, you'll probably use the `rpm -i kernel.rpm` command, and not `rpm -U kernel.rpm`. Installing (and not upgrading) newer kernels allows you to use the older kernel, in case the new kernel does not work for you.

Be careful. Install (`-i`), don't upgrade (`-U`) your new kernel. Otherwise, if you have a problem, you won't be able to go back to the old working kernel. Installing (`-i`) a new kernel with a command such as:

```
# rpm -i /tmp/kernel-2.4.21-4.1.EL.i386.rpm
```

installs the kernel, initial RAM disk, `System.map`, and config files automatically in the `/boot` directory. In addition, the RPM automatically adds a new stanza to your boot loader configuration file. For GRUB, the file is `/etc/grub.conf`.

 exam**Watch**

The `/etc/grub.conf` configuration file is linked to the actual GRUB configuration file, `/boot/grub/grub.conf`. You can open either filename in the text editor of your choice.

 exam**Watch**

To change the default boot stanza in GRUB, change the default variable. For example, if `default=0`, the default kernel loaded is the first stanza in `/boot/grub/grub.conf`. Similarly, if `default=1`, the default kernel loaded is the second stanza in `/boot/grub/grub.conf`.

Kernel Patches

Sometimes, all you need is a simple patch to a kernel. Patches usually work fairly well if you're upgrading from one patch version to the next higher version, such as from 2.4.21 to 2.4.22.

Kernel patches are easily available from Internet sites such as ftp.kernel.org. For example, if you want to upgrade from kernel version 2.4.21 to kernel version 2.4.22, download the `patch-2.4.22.gz` file from the Internet. Copy the patch to the `/usr/src` directory. Move to that directory, and run a command similar to the following to make the upgrade:

```
# zcat patch-2.4.22.gz | patch -p0
```

If it doesn't work, you'll see files with a `.rej` extension somewhere in your kernel source tree. Use a command such as **find** to check for such files. If you don't find any of these files, you can proceed with the **make clean**, **make menuconfig**, and **make dep** commands as described in the next section.



Generally, it may not be advisable to patch a Red Hat built kernel on your Red Hat systems. Different build versions of Red Hat Enterprise Linux kernels often incorporate changes that are part of generic Linux kernel patches.

mkbootdisk

The **mkbootdisk** utility creates a boot disk customized for your configuration. This is basically a rescue disk. After creating this disk, you can use it to simply boot your system, or you can type **rescue** at the boot prompt. Whenever you upgrade or install a new kernel, you should upgrade your boot disk. It's a simple command; in most cases, all you need to do is specify the version number associated with the desired kernel as follows:

```
# mkbootdisk 2.4.21-4.1EL
```

This command automatically takes the kernel with the specified version number from the `/boot` directory and writes it with appropriate configuration information to the first floppy drive, `/dev/fd0`. Table 5-2 lists several other options that may come in handy when using **mkbootdisk**.

Here is another example of the **mkbootdisk** command:

```
# mkbootdisk --device /dev/fd1 --verbose --noprompt 2.4.21-4.1.EL
```

This command creates a boot disk on the *second* floppy drive. It does not suppress output, nor does it prompt for a disk to be inserted. It uses kernel version 2.4.21-4.1.EL. When you boot Linux with this disk, you can use it as a rescue disk. Just type **linux rescue** at the boot prompt.

TABLE 5-2 mkbootdisk Command Switches

Command	Description
--device <i>device file</i>	Specifies where to put the image.
--mkinitrdargs <i>args</i>	Passes arguments to mkinitrd .
--noprompt	Won't prompt to insert a disk.
--verbose	Normally, mkbootdisk has no output. This option turns the output on.

on the
o b

By default, the first floppy drive on a PC is associated with /dev/fd0; the second floppy drive is associated with /dev/fd1.

exam
Watch

A significant portion of the practical exam tests your ability to recover a system that has failed in some way. While I am prohibited by the Red Hat non-disclosure agreement from providing the exact nature of the problems, I can say that being able to use rescue disks is a very important Linux administration skill.

Updating Your Boot Loader

If properly configured, the Red Hat kernel that you install should automatically update your boot loader. But as a RHCT or RHCE, you need to know how to check. If the code described in this chapter has not been added, you'll need to know how.

Whether you're using GRUB or LILO, it is advisable to keep your old kernel in case something goes wrong. So you'll be adding a stanza to either /etc/grub.conf or /etc/lilo.conf. In either case, the changes that you'll make will be as if you're setting up two different operating systems.

Updating GRUB

Look at your /etc/grub.conf file. If you have Linux on your system and use GRUB, you should already have a stanza that points to the appropriate locations for your original Linux kernel and Initial RAM disk. For example, here is an excerpt from my RHEL 3 /etc/grub.conf file (which includes a dual-boot configuration with Microsoft Windows):

```

title Red Hat Enterprise Linux ES (2.4.21-4.EL)
    root (hd0,0)
    kernel /vmlinuz-2.4.21-4.EL ro root=LABEL=/
    initrd /initrd-2.4.21-4.EL.img
title DOS
    rootnoverify (hd0,1)
    chainloader +1

```

In Red Hat Enterprise Linux, the `vmlinuz` and `initrd` files are already in the `/boot` directory. Since you've copied the revised kernels to the same directory, all you need is a second stanza that points to your revised files. When I revised my kernel earlier in this chapter, my `EXTRAVERSION` variable in `/usr/src/linux-2.4/Makefile` was `-4.ELcustom`. The changes are in bold:

```

title Red Hat Enterprise Linux ES (2.4.21-4.EL)
    root (hd0,0)
    kernel /vmlinuz-2.4.21-4.EL ro root=LABEL=/
    initrd /initrd-2.4.21-4.EL.img
title Red Hat Enterprise Linux ES (2.4.21-4.ELcustom)
root (hd0,0)
kernel /vmlinuz-2.4.21-4.ELcustom ro root=LABEL=/
initrd /initrd-2.4.21-4.ELcustom.img
title DOS
    rootnoverify (hd0,1)
    chainloader +1

```

Since you don't need to load `/etc/grub.conf` into the MBR, no further action is required. The resulting GRUB menu looks like Figure 5-10. Note how the original kernel is set as the default. If you've watched closely, you'll note that in `/etc/grub.conf`, the value of **default** was changed from 0 to 1. If you want to set the default to the new kernel, change the value of **default** back to 0.

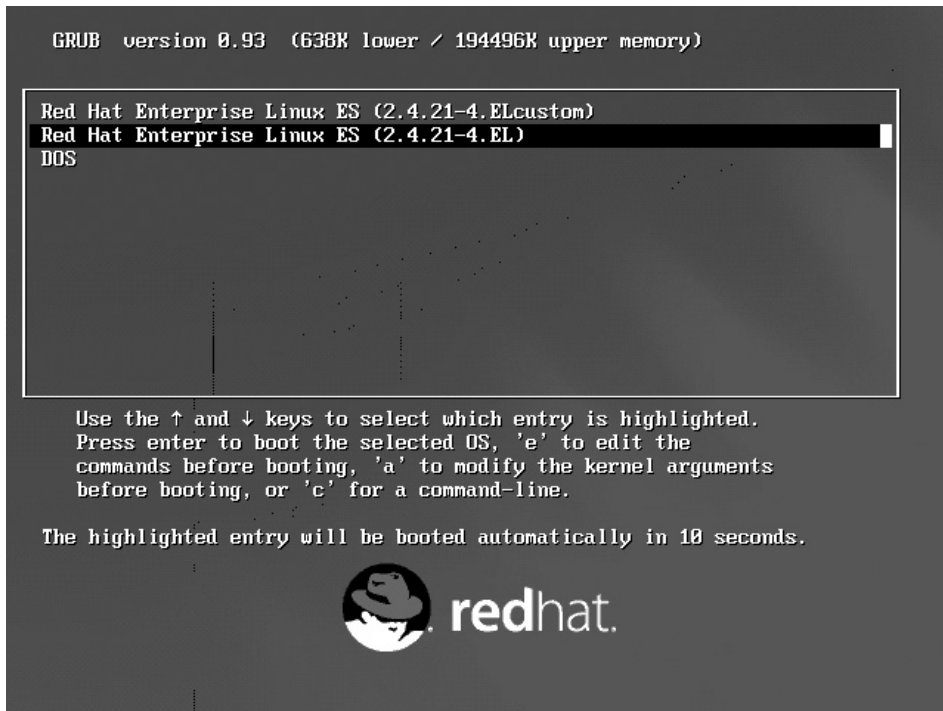
Updating LILO

Alternatively, if you're using LILO as a boot loader, you'll need to revise `/etc/lilo.conf`. Add a stanza that points to the new kernel. Take a look at the following excerpt from `/etc/lilo.conf`:

```

image=/boot/vmlinuz-2.4.21-4.EL
    label=linux
    initrd=/boot/initrd-2.4.21-4.EL
    read-only
    append="root=LABEL="/>

```

FIGURE 5-10 GRUB menu with original and recompiled kernels

From this information, you can see that the original kernel is called `vmlinuz-2.4.21-4.EL`. Assume LILO resides on the MBR and controls the boot process.

Now add another stanza for the new kernel.

```
image=/boot/vmlinuz-2.4.21-4.EL
label=linux
initrd=/boot/initrd-2.4.21-4.EL
read-only
append="root=LABEL=/"
image=/boot/vmlinuz-2.4.21-4.ELcustom
label=linux
initrd=/boot/initrd-2.4.21-4.ELcustom
read-only
append="root=LABEL=/"
```

Save this file, then run the `lilo -v` command. The output should resemble the following:

```
LILO version 21.4-4, Copyright (C) 1992-1998 Werner Almesberger  
'lba32' extensions Copyright (C) 1999,2000 John Coffman
```

```
Reading boot sector from /dev/hda  
Merging with /boot/boot.b  
Mapping message file /boot/message  
Boot image: /boot/vmlinuz-2.4.21-4.EL  
Mapping RAM disk /boot/initrd-2.4.21-4.EL  
Added linux *  
Boot image: /boot/vmlinuz-2.4.21-4.ELcustom  
Added newLinux  
Backup copy of boot sector in /boot/boot.0300  
Writing boot sector.
```

When you reboot, LILO will wait for you to enter a label, in this case, either linux or newLinux.

The icon consists of a grey triangle pointing downwards, with the word "exam" in a sans-serif font to its left and the word "Watch" in a bold, sans-serif font to its right.

Although there are references to both GRUB and LILO in the RHCT and RHCE exam curricula, Red Hat is focusing on GRUB, and has “deprecated” LILO. Therefore, I believe you can expect to work with GRUB on the Red Hat exams.

CERTIFICATION OBJECTIVE 5.05

Kernel Sources

One of the strengths of Linux is the ease with which you can customize your kernel to precisely meet your needs. But before you can start this process, you need the Linux kernel source code.

While references to recompiling the kernel have been removed from the Red Hat exam requirements, you may still need to find kernel modules and configuration files, which I cover in this section.

The icon consists of a grey triangle pointing downwards, with the word "exam" in a sans-serif font to its left and the word "Watch" in a bold, sans-serif font to its right.

Fortunately, the Red Hat exams no longer specify requirements to recompile the Linux kernel. Nevertheless, it is a very important skill for any Linux administrator.

The Kernel Source Tree and Documentation

Once installed, you can easily access the kernel source code through the `/usr/src/linux-2.4` directory. Installation procedures are addressed in following sections. Once the source code is installed, the `/usr/src` directory should look similar to the following:

```
# ls -l /usr/src/
total 12
drwxr-xr-x  2 root  root  4096 Jan 24  2003 debug
lrwxrwxrwx  1 root  root    17 Oct 23 16:12 linux-2.4 -> linux-2.4.21-4.EL
drwxr-xr-x 16 root  root  4096 Nov  7 15:56 linux-2.4.21-4.EL
drwxr-xr-x  8 root  root  4096 Nov  9 12:13 redhat
```

In this case, the physical directory is `/usr/src/linux-2.4.21-4.EL`, and there is a soft link called `linux-2.4` that points to this directory.

Periodically, you may wish to upgrade your kernel. If you use Red Hat sources, the new RPM packages should automatically update the symbolic link from `/usr/src/linux-2.4` to the real directory with the new kernel's source code (for example, symlink it to `linux-2.4.21-4.1.EL`). If necessary, you could perform the same action with the following command:

```
# ln -s linux-2.4.21-4.1.EL linux-2.4
```

The `/usr/src/linux-2.4` directory on my RHEL 3 computer includes the following files and directories:

arch	Documentation	kernel	README
configs	drivers	lib	REPORTING-BUGS
COPYING	fs	MAINTAINERS	Rules.make
COPYING.modules	include	MAKEFILE	scripts
CREDITS	init	mm	tmp_include_depends
crypto	ipc	net	

Begin your study of the current kernel with the `README` file. While the instructions in this chapter work with the current configuration of RHEL 3 on my computer, details can change from kernel to kernel. Also, examine the `Documentation` directory. It contains everything you need, from information on setting up symmetrical multiprocessors to serial consoles.

The other directories mainly contain source code, and you probably won't need to spend time examining those files (unless you *really* want to see how TCP/IP works). There is also a hidden file, `.config`, that may be present in this directory. I'll describe this file in more detail later in this chapter.

The Kernel RPMs

If you don't see the directories mentioned in the preceding section, then you haven't installed your Linux kernel's source code. There are two ways to install the source code for your RHEL 3 system. You could access the Red Hat RPMs from CD or a network source and install the kernel RPM. Alternatively, you could install the packages through the **redhat-config-packages** utility. The following example is based on an installation from the appropriate Red Hat Installation CD:

```
# mount /mnt/cdrom
# rpm -Uvh /mnt/cdrom/RedHat/RPMS/kernel-source-*
```

Depending on the packages you've specified when you installed RHEL 3, this may be all you need. However, this command may bring up some error messages similar to the following:

```
error: Failed dependencies:
      gcc >= 2.96-98 is needed by kernel-source-2.4.21-4.EL
```

which suggests that you need to install some other RPM packages, as described in the next section.

Required RPMs

In order to build a kernel from the source code, you need to ensure you have all the RPMs necessary, not only for the kernel, but also for the tools needed to build the kernel. You can check your system to ensure you have the RPM packages described in Table 5-3. Many of these packages have a number of other dependencies.

If not, mount the appropriate Red Hat installation CDs or network installation source and install the needed packages with the **rpm -Uvh *packagename*** command. If the revision associated with your package and architecture are different, revise the package names accordingly.

The packages in Table 6-3 end with an *, because version numbers change frequently. In any case, the objective is to install these packages; for example, to install the tk package from a mounted CD, all you need is the **rpm -Uvh /mnt/cdrom/RedHat/RPMS/tk-*** command.

However, you'll also need to install any related dependent packages. For example, the kernel-source-* RPM requires the appropriate GNU C Compiler, which is the gcc-* RPM.

TABLE 5-3

Kernel Source RPMs

Package	Description
kernel-source-*	Kernel source files
glibc-kernheaders-*	Kernel header files
glibc-devel-*	Required for C libraries
cpp-*	C language preprocessor
ncurses-*	Required for menuconfig screen
ncurses-devel-*	Development libraries for ncurses
binutils-*	Required binary utilities
gcc-*	C language compiler
tcl-*	TCL scripting language—required for xconfig screen
tk-*	TK X Window widgets—required for xconfig screen



*Wildcards such as an * are often also known as globbing.*

GUI Kernel Source Management

It can take time to load all of the RPMs required to compile and customize the Linux kernel. It can be difficult to remember to install every RPM. There is an alternative: the Red Hat Package Management utility. I'll show you how to use this tool to load the kernel source packages in Exercise 5-3.

EXERCISE 5-3

Loading the Kernel Source Graphically

In this exercise, we'll install the source code and associated files for the Linux kernel using the Red Hat Package Management utility. I'll assume that you're using the same network source that you created in Chapter 2. This exercise assumes that network source is on a remote computer with an IP address of 192.168.30.4. Substitute the IP

address (or if you have a DNS server for your LAN, the hostname) for your network source computer. You'll need to start from the Linux GUI.

1. If you're not already in the Linux GUI, run the **startx** command.
2. Open a command line interface. Right-click on the desktop, and click New Terminal in the pop-up menu that appears.
3. At the command line interface, make a connection to the network source. I'm assuming that you have an active NFS connection, no firewall, and a local `/mnt/inst` directory. For the given source, run the following command (substitute your IP address or hostname, and preferred mount directory in this command):

```
# mount -t nfs 192.168.30.4:/mnt/inst /mnt/inst
```

4. Start the Package Management utility. Point to the mounted network source (without the `--tree` option, the Package Management utility works from the installation CDs):

```
# redhat-config-packages --tree=/mnt/inst
```

5. Select the Kernel Development package group for installation.
 6. Click Update. The Package Management utility computes the dependencies, adding RPMs to the list of packages to be installed.
 7. Once you see the Completed System Preparation window, click Continue.
 8. The Package Management utility starts installing the packages you need to examine the source code and recompile the Linux kernel.
 9. When installation is complete, click OK.
 10. Click Quit to exit from the Package Management utility.
-

The Linux Kernel tar File

Alternatively, you can download the newest kernel from the Linux kernel home page at www.kernel.org. The version numbers are discussed in the next section. Once you have downloaded the kernel source, you will need to properly install it. For our example, we will assume you downloaded `linux-2.4.22.tar.gz` into the `/usr/src/` directory.

```
# cd /usr/src
# mkdir linux-2.4.22
```

```
# rm linux-2.4
# tar xzvf linux-2.4.22.tar.gz
# ln -s linux-2.4.22 linux-2.4
```

Here we manually created a new directory for the kernel. Then we removed the old link, and then uncompress the tar.gz file. Then you can link linux-2.4 to the new directory created when you uncompress the tar.gz file.



Compressed tar files are shown in tar.gz format; they are also known as “tarballs.”

CERTIFICATION OBJECTIVE 5.06

Recompiling a Kernel

Fortunately, references to recompiling the Linux kernel have been removed from the Red Hat exam requirements. However, RHCEs in the real world are expected to know how to perform high-level tasks such as optimizing and recompiling the Linux kernel.

In this section, we'll start by looking at the kernel configuration file. Then we'll proceed with the different tools available to edit the kernel configuration. Finally, I'll show you the commands needed to compile your new configuration into the kernel, and show you the files this adds to the /boot directory and the settings it adds to the boot loader.



References to recompiling the Linux kernel have been removed from the Red Hat Exam Prep guide and associated Red Hat prep courses. However, good Linux administrators know how to recompile the Linux kernel.

The Kernel Configuration Scripts

After you've configured a kernel once, the configuration information is stored in a hidden file, /usr/src/linux-2.4/.config. It is structured as a listing of variables. Here are some entries from the .config file:

```
CONFIG_NETDEVICES=y
CONFIG_DUMMY=m
# CONFIG_SUNLANCE is not set
```

As you can see, there are three main types of variables in this file. The first command here compiles in direct support, the second entry compiles support as a module (the “m”), and the third is commented out, and is therefore not compiled into the kernel at all. You should never have to edit this file directly, as there are easier ways to configure your kernel.

Move to the directory with your kernel source files. If you’ve installed the RHEL 3 kernel-source RPM, you can use the `/usr/src/linux-2.4` directory. If you’ve installed the latest kernel from `www.kernel.org`, it may be in the `/usr/src/linux` directory. Three tools can help you configure the kernel configuration file: **make config**, **make menuconfig**, and **make xconfig**.

Back Up Your Configuration

The default configuration for your current Linux kernel is stored in the `/boot` directory. For the default RHEL 3 system, it’s stored in the `config-2.4.21-4.EL` file. Back up this file on another location such as a rescue floppy so that you can restore your current kernel configuration if all else fails.

You can use your current configuration as a baseline; the Linux kernel configuration tools will start with these settings. To do so with the current kernel, run the following command:

```
# cp /boot/config-2.4.21-4.EL /usr/src/linux-2.4/.config
```

Alternatively, there are a number of standard configuration files in the `/usr/src/linux-2.4/configs` directory. If you want to start with one of these files, use the config file that corresponds most closely to your hardware. You can set that as the starting point for your configuration by copying it to the `/usr/src/linux-2.4/.config` file.



There is a default kernel configuration file in the `/usr/src/linux-2.4/configs` directory. It’s associated with your CPU hardware; for example, if you have an Athlon CPU, you’ll find it in a file named `kernel-2.4.21-athlon.config`. If your kernel version is different, the number changes accordingly.

make config

Once you’re in the directory with the kernel source files (`/usr/src/linux-2.4`), you can call a simple script to configure a new kernel with the following command:

```
# make config
```

This script will prompt you through your different options. Figure 5-11 shows an excerpt from the output for this script.

FIGURE 5-11 Questions from the **make config** utility

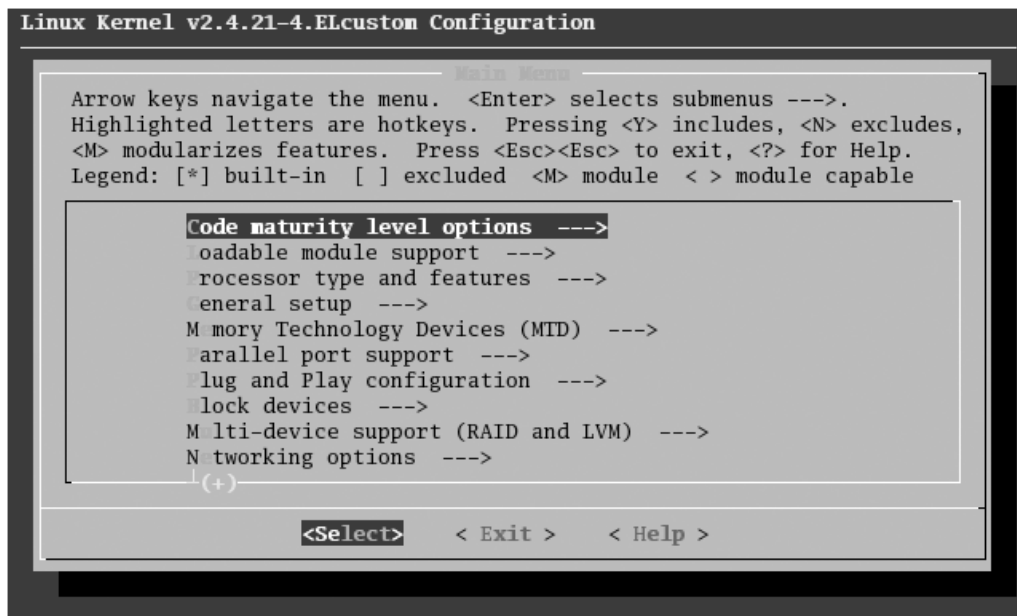
```
( cd include ; ln -sf asm-i386 asm)
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?]
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?]
  Set version information on all module symbols (CONFIG_MODVERSIONS) [Y/n/?]
  Kernel module loader (CONFIG_KMOD) [Y/n/?]
*
* Processor type and features
*
Processor family (386, 486, 586/K5/5x86/6x86/6x86MX, Pentium-Classic, Pentium-MMX, Pentium-Pro/Celeron/Pentium-II, Pentium-III/Celeron(Coppermine), Pentium-4, K6/K6-II/K6-III, Athlon/Duron/K7, Opteron/Athlon64/Hammer/K8, Elan, Crusoe, Winchip-C6, Winchip-2, Winchip-2A/Winchip-3, CyrixIII/VIA-C3, VIA-C3-2) [Pentium-Pro/Celeron/Pentium-II]
```

Here the kernel variables are listed in parentheses and the possible answers are in brackets. The default answer is in capital letters. If you type in a `?`, you will see a help page explaining this option. Since several hundred questions are associated with this script, most administrators use one of the other two scripts to manage their Linux kernels.

make menuconfig

A nicer way to create the `.config` file is to use the **make menuconfig** command. This requires the `ncurses RPM` package. This opens a text-based menu-driven system that classifies and organizes the changes that you can make to a kernel. Figure 5-12 illustrates the main menu associated with this command.

The nice thing about **menuconfig** is that it works very nicely over a remote text connection from other Linux computers (or perhaps that is dangerous!). Also, options appear at the bottom of the menu to load or save the configuration file from a different location.

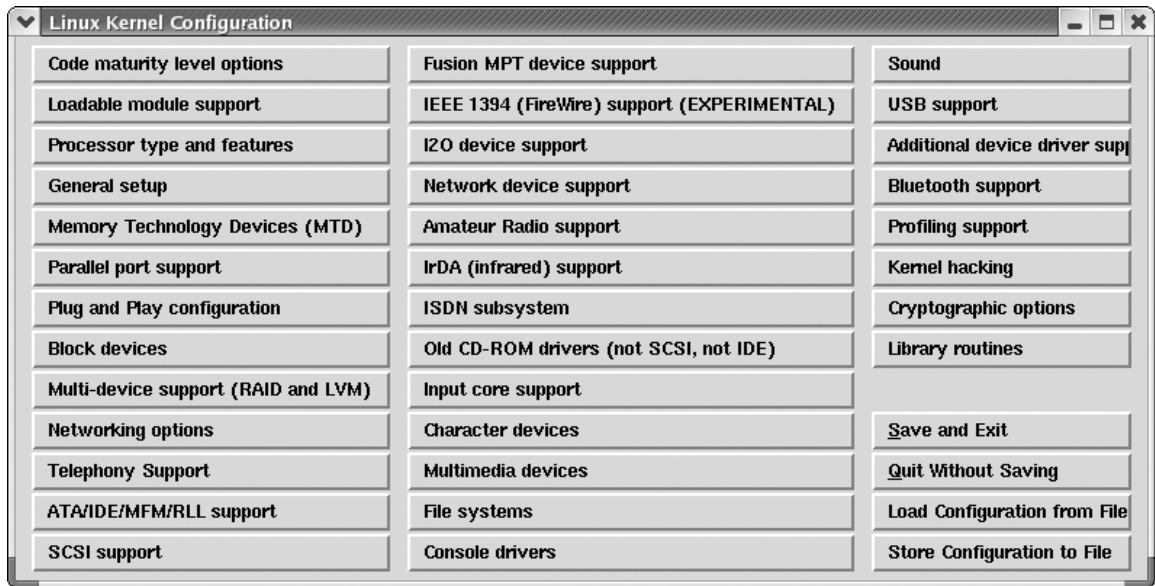
FIGURE 5-12 The `make menuconfig` configuration menu

make xconfig

The other way to make changes to the kernel is to use the X Window System. You can generate a graphical menu system to configure your kernel by running the `make xconfig` command. Figure 5-13 shows the `xconfig` main menu. You can also use `xconfig` to load or save the configuration from the file of your choice.

While this menu may look slightly different in different Linux distributions, the principles and basic options remain fairly constant. Each of the Kernel Configuration Options discussed on the following pages are presented for completeness. It is important for a Linux administrator to have a detailed understanding of the hows and whys about reconfiguring and recompiling the kernel.

You may not see any requirement to recompile the Linux kernel during your RHCE exam. Red Hat does not have to test all requirements listed in the RHCE Exam Prep guide. However, if you have to recompile the Linux kernel on the RHCE exam, you'll need to work quickly.

FIGURE 5-13 The make xconfig configuration menu

Understanding Kernel Configuration Options

To configure a kernel, you need to understand some of the main kernel configuration options. Each of the aforementioned kernel configuration tools includes help menus for just about every available option.

The Basic Kernel Configuration

If you're asked to change a setting such as `CONFIG_EXPERIMENTAL`, you don't have to search through different menus. You can find the governing menu using the information in the `/boot/config-2.4.21-4.EL` configuration file. As you can see in the file:

```
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
```

The `CONFIG_EXPERIMENTAL` variable can be found under the Code Maturity Level Options menu.

The Standard Red Hat Kernel Configuration

The standard RHEL 3 Linux kernel supports a wide variety of hardware and software. Almost every module that could be made is made. This is a big kernel, and numerous modules can be used for it with the standard installation. This is not a problem when you install RHEL 3, but it is highly recommended you streamline the standard kernel and remove unwanted modules.

In the following sections, I describe the different kernel configuration menus, section by section. However, since it appears that recompiling the kernel is no longer a requirement on the Red Hat exams, I do not provide any screenshots of additional menus. If you're interested in this level of detail, follow along on your Red Hat computer.

Code Maturity Level Options

The Code Maturity Level options allow you to incorporate experimental code in the kernel. Common examples include drivers for new hardware, esoteric filesystems, and network protocols. Experimental code is often also known as alpha level software. If you have obsolete code that you want to incorporate into your kernel, it also falls in this category, as newer kernels often omit support for older features.

The one option relates to the `CONFIG_EXPERIMENTAL` variable. It's enabled by default in RHEL 3.

Loadable Modules Support Options

The Loadable Module Support screen allows you to enable loadable modules. The kernel module loader will automatically load modules for most new hardware, when detected. As discussed earlier, loadable modules allow you to optimize the kernel. If you want to optimize the kernel, keep all of the options on this screen active.

Processor Type and Features

The Processor Type and Features menu in RHEL 3 supports a wide variety of CPUs.

General Setup Options

The General Setup Options menu includes some basic hardware and kernel configuration options. Many of these options are self-explanatory, and the defaults are generally acceptable. If you need more information, click the Help button associated with a specific kernel option.

On a network, you want networking support. Most computers have PCI cards, and the defaults give you full PCI support, using BIOS detection, documenting the detected cards in the `/proc` directory. Various types of hot-pluggable devices are now popular, including PCMCIA cards for laptops and PCI hotplug support for removable drives.

The System V IPC, BSD Process Accounting, and Sysctl support parameters are all standard for current kernels. While Linux kernels are normally associated with ELF binaries, the other binaries may help with certain programs.

Memory Technology Devices

The options shown in the Memory Technology Devices menu allow you to set up Linux for basic “Flash” memory cards, including those that might be installed through a PCMCIA adapter. Unless you’re planning to use some of these devices in the future, keep this option disabled.

Parallel Port Support

The options shown in the Parallel Port support menu are based on hardware that may be connected to your computer through a parallel port. This includes everything from printers through parallel port hard drives. Remember that it is normally best to avoid the Experimental options unless you are a developer working on supporting the associated drivers.

Plug and Play Configuration

The options shown under the Plug and Play Configuration menu activate basic plug and play support on your Linux computer. Generally you should keep the defaults. While Linux plug and play does not handle all ISA and PCI devices, it does help you configure your computer for Linux.

Block Device Options

Under the Block Device Options menu, you can specify your floppy devices and nonstandard hard disks. You can specify support for parallel port ATAPI CD-ROMs, tape drives, and even ATAPI floppy drives. You can also enable loopback support and network block support (which lets you use a physical disk on the network as if it were a local disk). If you have any parallel port devices such as external CD-ROMs or hard drives, you could enable support for those here. You can also set up support for RAM disks under this menu.

Multidevice Support for RAID and LVM

If you're ever going to set up a RAID array of disks to help protect your data, you can enable that option in the Linux kernel under the Multidevice Support for RAID and LVM menu. If you ever want to put together a volume set, where a directory can span more than one partition on more than one physical hard disk, you can enable that option here as well.



While there is support for RAID-4 in the Linux kernel, it is not directly supported by RHEL 3 or currently available versions of Red Hat or Fedora Linux.

Networking Options

There are many options for networking in the Linux kernel. The Networking Options menu is extensive. It includes a substantial number of settings for different hardware devices, software settings, and more.

Telephony Support

Telephony support on a computer network uses special network cards to convert voice into the type of data that can be sent over a network. Linux offers some limited telephony support through the kernel as shown under the menu of the same name.

ATA/IDE/MFM/RLL Support

These acronyms all relate to various types of regular PC hard disk and CD drive interfaces. Normally, you shouldn't disable this kernel option unless all hard disk storage on your system is based on a SCSI interface. Even then, the flexibility of being able to install IDE devices is usually worth the extra code this adds to the kernel.

SCSI Support Options and Low-Level Drivers

You can enable SCSI hard disks, tape drivers, and CD-ROM support in this section. If you have a SCSI CD-ROM jukebox, or any other device that requires more than one SCSI Logical Unit Number (LUN), you may have to enable probing of all LUNs.

Near the bottom of the menu, you can configure verbose SCSI error reporting. You can also enable specific low-level SCSI support. Red Hat includes support for high-end hardware RAID-enabled SCSI host adapters, including 64-bit PCI adapters.

Fusion MPT Device Support

This menu supports modules associated with very high speed SCSI adapters, associated with hardware developed by LSI logic.

IEEE 1394 Support

The IEEE 1394 standard is more popularly known as FireWire or iLink. It's basically a very high speed hot plug and play hardware option, with data transfer speeds in the hundreds of Mbps. Linux support for IEEE 1394 standards is far from complete. Kernel support for any IEEE 1394 device is currently officially experimental. However, support for IEEE devices such as external hard drives are readily available, and configured as modules by default in RHEL 3.

I2O Device Support

The I2O specification, also known as Intelligent I/O, supports split drivers which can optimize communication performance between a device and the rest of your computer. Don't enable I2O haphazardly; it requires hardware that supports it.

Network Device Support

Linux supports a wide range of network cards. The Network Device Support menu allows you to enable support for the adapters you may need. Generally, you should enable support for only network devices that you're using now or may use in the future.

Amateur Radio Support

Linux supports connections to various amateur radios. Unless you plan to connect your computer to an amateur radio station in the future, there is no need to enable support for any of these devices.

IrDA Support

Linux supports Infrared connections, mostly for network support. The IrLAN protocol supports wireless access points. The IrNET protocol requires PPP. The IrCOMM protocol sets up port emulation, useful for setting up terminals and printers. For a list of supported infrared-port device drivers, click that button and activate the devices that you need.

ISDN Options

Integrated Services Digital Networks (ISDN) lines are a fairly popular high-speed digital option, especially outside of North America. Adding ISDN support allows you to use an ISDN card for inbound or outbound dialing connections. The ISDN device has a built-in AT-compatible modem emulator, autodial, channel-bundling, callback, and caller authentication without the need for an external daemon to be running. Under the ISDN Options menu, you can enable synchronous Point-to-Point Protocol (PPP) connections. The supporting `isdn4k-utils` RPM is installed by default on RHEL 3.

Older CD-ROM Support Options

If you have an older CD-ROM that is not an IDE or SCSI CD-ROM, then you need to enable special support for it in the Linux kernel, as shown when you click the “Old CD-ROM drivers (not SCSI, not IDE)” option. This section has many drivers for Mitsumi, Goldstar, Philips, Sony, Sound Blaster, and other old CD-ROM and disk types.

Input Core Support

The Input Core Support section configures support for various basic input devices: keyboards, mice, and joysticks. These devices are modular by default, which allows Linux to recognize these basic devices using plug and play detection.

Character Device Options

Character devices send their data in byte streams. Typical character devices range from serial ports to virtual consoles. The Character Devices submenu allows you to specify support for a wide variety of devices, including virtual terminals, serial ports, newer AGP video cards, mice, joysticks, non-SCSI tape drives, and more.

Multimedia Devices

The Multimedia Devices options support a wide range of video capture and AM/FM radio devices. Click each option (Video for Linux, Radio Adapters) for a list of drivers which you can enable. As always, it is best to keep what you enable to a minimum.

Filesystem Options

The File Systems subsection is a list of all the different types of filesystems Linux supports. Select the Quota option if you need to support quotas. You can also compile in the kernel Automounter to support remote filesystems.

Because Linux supports so many different hardware platforms, it includes support for a large number of filesystem types. However, because of the proprietary nature of some filesystems, the degree of support is variable. You'll note that support for a lot of filesystems in this menu is experimental; in fact, support for writing to NTFS filesystems may be dangerous!

Console Driver Options

Linux supports console drivers, which can set up text on most graphics systems, even when Linux doesn't detect the right cards or monitors. The Frame Buffer Support option supports video adapters that store images in frame buffers.

Sound System Support Options

A wide variety of sound cards are supported by RHEL 3, normally as modules. These cards range from the Ensoniq Audio PCI card to TV card mixers. You can also use these drivers for cards that emulate the appropriate hardware. Check the Sound submenu for the latest list of supported hardware for your kernel. If you have a card not named in the previous list, try to see if it emulates any card on the list. Many proprietary cards do emulate products from Sound Blaster or offer OPL/2 or OPL/3 compatibility.

USB Support

Linux supports a number of USB mass storage devices, printers, input devices, cameras and scanners, and even modems. Linux support for USB networking cards is still officially experimental. Linux support for USB is improving, though some USB drivers that you'll see in the USB support menu are still considered experimental. Unfortunately, this includes support for faster USB 2.0 standard drivers.

Additional Device Driver Support

As befits the name, the Additional Device Driver support menu allows you to configure the Linux kernel for a variety of hardware devices. These devices range from Gigabit Ethernet adapters to Storage Area Network devices.

Bluetooth Support

Bluetooth is a radio technology for short-range networks. You can configure a number of Bluetooth devices in the Linux kernel.

Profiling Support

Profiling support allows you to use the OProfile system to characterize the performance of your system. It is described in more detail at oprofile.sourceforge.net.

Kernel Hacking

Kernel hacking allows you to use the drivers you need to debug driver or related Linux kernel issues.

Cryptographic Options

The Cryptographic options supports software associated with strong encryption in Linux. It's disabled by default in RHEL 3.

Library Routines

The Library Routines support compression in Linux. Unlike for Red Hat Linux 9, they are enabled by default for RHEL 3.

Compiling and Installing a Custom Kernel

After setting up all the options you want from one of the Kernel configuration programs, make sure to save your changes. The changes are saved in the appropriate .config file. Next, you can compile your kernel. Compiling a new kernel can be a long and tedious process. The following is a list of things you should do to successfully compile your kernel—in order:

1. If you haven't already done so, create a custom boot floppy disk with the following command. Test the floppy that's created.

```
# mkbootdisk `uname -r`
```
2. Run the `cd /usr/src/linux-2.4` command to reach the proper directory. You'll need to run your commands to modify and compile the kernel in this directory. This assumes that you've installed the appropriate kernel-source RPM. In RHEL 3, this directory is linked to the directory with the source code, `/usr/src/linux-2.4.21-4.EL`.
3. Open `/usr/src/linux-2.4/Makefile` in a text editor. In RHEL 3, the `EXTRAVERSION` variable is already set to identify a custom kernel.

If necessary, change the value of this variable to uniquely identify your kernel. Record the value of this variable.

```
EXTRAVERSION= -4.ELcustom
```

4. Save your current kernel configuration file. It's stored in the `/boot` directory. For RHEL 3, it's the `/boot/config-2.4.21-4.EL` file. Have it ready with a rescue floppy in case disaster strikes. Copy it to the `/usr/src/linux-2.4/.config` file.
 - A. Alternatively, you can start from a standard configuration file in the `/usr/src/linux-2.4/configs/` directory. Alternatively, you can run the **make oldconfig** command.
5. Run the **make mrproper** command to ensure your source files are in a consistent and clean state.
6. Now you can customize your kernel with one of the following three utilities: The **make config** command runs the text-based configuration script. The **make menuconfig** command runs a low-level graphical configuration script. The **make xconfig** command starts the X Window System configuration tool.
7. Make any changes as required by your hardware and or the RHCE exam. Before you exit from the kernel configuration utility, make sure to save your changes!
8. The **make dep** command will set up all your dependencies correctly. This takes the settings from `/usr/src/linux-2.4/.config` and pushes them down into the correct source code subdirectory. This will take some time, as there are nearly 1,500 possible kernel settings on a RHEL 3 system!



The make dep process took about 20 minutes on my computer with a 1.2GHz CPU.

9. Run the **make clean** command to prepare the source code directories to build your new kernel.
10. The **make bzImage** command compiles the kernel and creates a gzip compressed kernel image file.



The make bzImage process took about 20 minutes on my computer with a 1.2GHz CPU.

11. Now you need to build your modules (you did include kernel module support, didn't you?). Use **make modules** to build all your modules.



The make modules process took about 50 minutes on my computer with a 1.2GHz CPU.

- 12. Now you need to install your newly made modules, which will install all your loadable modules into a new subdirectory of the `/lib/modules/kernelversion` directory. The *kernelversion* should correspond to the EXTRAVERSION variable in your Makefile. Use **make modules_install** to install your modules.
- 13. With our kernel now built, we need to move it to `/boot` before we can boot from it. Run the **make install** command to copy the new kernel and create a new Initial RAM Disk file (`initrd`) in the `/boot` directory. It also updates the default GRUB boot loader.



*The make install command copies your new kernel from the `/usr/src/linux-2.4/arch/i386/boot` to the `/boot` directory. It also uses the **mkinitrd** command to create a new Initial RAM disk in the `/boot` directory.*

- 14. Verify the changes to the `/boot` directory.

Building a kernel is an involved process, but it follows a standard pattern. It is very important that you become familiar with kernel construction procedures and troubleshooting. Refer to the following Scenario and Solution for some common problems encountered and their recommended solutions.

SCENARIO & SOLUTION

You looked under <code>/usr/src/</code> but did not see the Linux kernel source code. What did you do wrong?	You did not install the kernel source code. Install the kernel-source and related RPMs or use the Red Hat Package Management utility to install the Kernel Development package group.
You configured the kernel as a monolithic kernel, but when you run make bzImage , it fails, saying the kernel is too big. What should you do?	You must reconfigure your kernel to use modules. There are limits on the size of a compressed kernel, so you should always build a modular kernel.
You can't find your new kernel.	Did you create an appropriate EXTRAVERSION variable in your Makefile? Did you use this variable throughout this process? If not, then you may not be able to correctly locate the kernel or its modules.

CERTIFICATION OBJECTIVE 5.07

The cron and at Systems

The cron system is essentially a smart alarm clock. When the alarm sounds, Linux runs the commands of your choice automatically. You can set the alarm clock to run at all sorts of regular time intervals. Alternatively, the at system allows you to run the command of your choice, once, at a specified time in the future.

Red Hat configured the cron daemon, `crond`. By default, it checks a series of directories for jobs to run, every minute of every hour of every day. The `crond` checks the `/var/spool/cron` directory for jobs by user. It also checks for scheduled jobs for the computer under `/etc/crontab` and in the `/etc/cron.d` directory.

The behavior of the Linux cron is different from under Unix, where the cron daemon wakes up only when it needs to launch a program.



Because cron always checks for changes, you do not have to restart cron every time you make a change.

The System crontab and Components

The crontab file is set up in a specific format. Each line can be blank, a comment (which begins with `#`), a variable, or a command. Blank lines and comments are ignored.

When you run a regular command, the actions of the shell are based on environmental variables. To see your environmental variables, run the `env` command. Some of the standard variables in Red Hat Enterprise Linux include: `HOME` as your home directory, `SHELL` as the default shell, and `LOGNAME` as the username.

You can set different variables within the crontab file, or you can set environmental variables with the following syntax:

```
Variable=Value
```

Some variables are already set for you. For example, `HOME` is your home directory, `SHELL` is the user's default shell, and `PATH` is where the shell looks for commands. You can set these variables to different values in your crontab file. For example, the default `/etc/crontab` file includes the following variables:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

```
MAILTO=root
HOME=/
```

Note that the values of PATH, MAILTO, and HOME are different from those for the standard environment variables.

on the
Job
exam
Watch

The MAILTO variable can help you administer several Linux systems. The cron daemon sends output by e-mail. Just add a line like MAILTO=me@somewhere.com to route all cron messages associated with that file to that e-mail address.

Note how the PATH variable in a crontab may be different from the PATH variable associated with your shell. In fact, the two variables are independent. Therefore, you'll want to know the exact path of every command in your crontab. Specify the absolute path with the command if it isn't in the crontab PATH.

Here is the format of a line in crontab. Each of these columns is explained in more detail in Table 5-4.

```
#minute, hour, day of month, month, day of week, command
* * * * * command
```

If you see an asterisk in any column, cron runs that command for all possible values of that column. For example, an * in the minute field means that the command is run every minute during the specified hour(s). Take another example, as shown here:

```
1 5 3 4 * ls
```

TABLE 5-4

Entries in a
crontab
Command Line

Field	Value
Minute	0-59
Hour	Based on a 24-hour clock; for example, 23 = 11 P.M.
Day of month	1-31
Month	1-12, or jan, feb, mar, etc.
Day of week	0-7; where 0 and 7 are both Sunday; or sun, mon, tue, etc.
Command	The command you want to run

This line runs the **ls** command every April 3, at 5:01 A.M. The asterisk in the day of week column simply means that it does not matter what day of the week it is; crontab still runs the **ls** command at the specified time.

The crontab file is flexible. For example, a 7-10 entry in the hour field would run the specified command at 7:00 A.M., 8:00 A.M., 9:00 A.M., and 10:00 A.M. A list of entries in the minute field such as: 0,5,10,15,20,25,30,35,40,45,50,55 would run the specified command every five minutes. The cron daemon also recognizes abbreviations for months and the day of the week.

The actual command is the sixth field. You can set up new lines with a percent (%) symbol. This is useful for formatting standard input. The example that follows formats input for an e-mail message. Here is an example cron file:

```
# crontab -l
# Sample crontab file
#
# Force /bin/sh to be my shell for all of my scripts.
SHELL=/bin/sh
# Run 15 minutes past Midnight every Saturday
15 0 * * sat $HOME/scripts/scary.script
# Do routine cleanup on the first of every Month at 4:30 AM
30 4 1 * * /usr/scripts/removecores >> /tmp/core.tmp 2>>&1
# Mail a message at 10:45 AM every Friday
45 10 * * fri mail -s "Project Update employees%Can I have a status
update on your project?%%Your Boss.%
# Every other hour check for alert messages
0 */2 * * * /usr/scripts/check.alerts
```

exam Watch

Take a look at some of the scripts in the */etc/cron.daily* directory. Three key scripts include **logrotate**, for rotating log files; **slocate.cron**, which updates the **locate** file database; and **tmpwatch**, which wipes files from */tmp* and */var/tmp* after a specific amount of time.

Setting Up cron for Users

Each user can use the **crontab** command to create and manage cron jobs for their own accounts. There are four switches associated with the **crontab** command:

- **-u user** Allows the root user to edit the crontab of another specific user.
- **-l** Lists the current entries in the crontab file.
- **-r** Removes cron entries.
- **-e** Edits an existing crontab entry. By default, crontab uses vi.

If you want to set up cron entries on your own account, start with the **crontab -e** command. It opens the vi editor, where you can add the variables and commands of your choice, similar to what you've seen in `/etc/crontab`.

EXERCISE 5-4

Create a cron Job

In this exercise, we will modify the basic Red Hat cron job settings to read a text file at 1:05 P.M. every Monday in the month of January. To do so, you'll need to create a directory for yearly cron jobs. To do this, use the following steps:

1. Log in as the root user.
 2. Create a `/etc/cron.yearly` directory. Add a file called `taxrem`, which reads a text file from your home directory. A command such as the following in the `taxrem` file should suffice:

```
cat ~/reminder
```
 3. Make sure to add appropriate lines to the `reminder` file in your home directory.
 4. Add an appropriate command to your `/etc/crontab` file. Based on the conditions described, it would read as follows:

```
5 13 * 1 1 root run-parts /etc/cron.yearly
```
 5. Save and exit.
-

Running a Job with the at System

Like cron, the `at` daemon supports job processing. However, you can set an `at` job to be run once. Jobs in the cron system must be set to run on a regular basis. The `at` daemon works in a way similar to the print process; jobs are spooled in the `/var/spool/at` directory and run at the specified time.

You can use the `at` daemon to run the command or script of your choice. For the purpose of this section, assume that user `michael` has created a script named `7e7` in

his home directory to process some airplane sales database to another file in the same directory called sales.

From the command line, you can run the **at time** command to start a job to be run at a specified *time*. That *time* can be now, in a specified number of minutes, hours, or days, or at the time of your choice. I illustrate several examples in Table 5-5.

You can use one of the example commands shown in Table 5-5 to open an at job. It opens a different command line interface, where you can specify the command of your choice. Assume you're about to leave work, and want to start the job in an hour. From the conditions specified above, you'd run the following commands:

```
$ at now + 1 hour
at> /home/michael/7e7 > /home/michael/sales
at> Ctrl-D
```

The CTRL-D command exits the at command shell and returns to your original command line interface. To check the status of your jobs, so you can see if it will work, run the following job queue command:

```
$ atq
1          2004-4-12 17:18 a michael
```

If there's a problem with the job, you can remove it with the **atrm** command. For the output shown, you'd remove job number 1 with the following command:

```
$ atrm 1
```

TABLE 5-5 Examples of the at Command

Time Period	Example	Description
Minute	at now + 10 minutes	Associated jobs will start in 10 minutes.
Hour	at now + 2 hours	Associated jobs will start in 2 hours.
Days	at now + 1 day	Associated jobs will start in 24 hours.
Weeks	at now + 1 week	Associated jobs will start in 7 days.
n/a	at teatime	Associated jobs will start at 4:00 P.M.
n/a	at 3:00 6/13/04	Associated jobs will start on June 13, 2004, at 3:00 A.M.

Securing cron and at

You may not want everyone to be able to run a job in the middle of the night. If there is a security flaw in your system, someone may download important data or worse, and it could be days before you discover the security breach.

As with network firewalls, you can allow or deny users the privilege of using cron. You can set up users in `/etc/cron.allow` and `/etc/cron.deny` files. If these files don't exist, cron usage is not restricted. If users are named in `/etc/cron.allow` file, all other users won't be able to use cron. If there is no `/etc/cron.allow` file, only users named in `/etc/cron.deny` can't use cron.

These files are formatted as one line per user; if you have the following entries in `/etc/cron.deny` and no `/etc/cron.allow` file, users `elizabeth` and `nancy` aren't allowed to set up their own cron commands:

```
elizabeth
nancy
```

You can secure access to the `at` system in the same way. The corresponding security configuration files are `/etc/at.allow` and `/etc/at.deny`.

CERTIFICATION SUMMARY

You can have great control over how your Linux installation is set up and configured. You can control almost all aspects of the user environment, as well as the details of your kernel. Any variables or system-wide functions you may need to run can be kept in the `/etc/bashrc` or `/etc/profile` script.

You can set up quotas to limit the user's disk usage. You can set up one quota per partition, and set soft and hard limits for users. With grace periods, you can set up a soft limit to give users an appropriate warning.

The easiest way to update a kernel is to install (and not upgrade it) from a Red Hat RPM. When you do, it automatically updates your boot loader files as needed.

Alternatively, the kernel can be optimized for your particular installation and hardware, and you have detailed control over its configuration. To make a modular kernel, you need to run six commands: **make mrproper**, **make config**, **make dep**, **make clean**, **make modules**, **make modules_install** and **make install**. If you are compiling a monolithic kernel, you will *not* need to run the last two module-related commands.

The `cron` and `at` daemons allows you to schedule jobs to run at any given time.



TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 5.

Shell Configuration Files

- ☐ All system-wide shell configuration files are kept in the `/etc` directory.
- ☐ `/etc/profile` is the system-wide startup shell script for bash users.
- ☐ All users have hidden shell configuration files in their home directories.

Setting Up and Managing Disk Quotas

- ☐ Quotas are used to limit a user's or a group of users' ability to consume disk space.
- ☐ Quotas are set on specific filesystems mounted to standard Linux formats.
- ☐ Quota support must be enabled in the kernel. By default, quotas are enabled in RHEL 3 kernels.
- ☐ Quotas have soft limits and hard limits. If both soft and hard limits are set, then a user can exceed his or her soft limit for a modest period of time.
- ☐ Users and groups may never exceed their hard limits.

The Basics of the Kernel

- ☐ The kernel lies at the heart of the operating system.
- ☐ You can recompile Linux kernels to minimize size and maximize efficiency.
- ☐ To optimize the Linux kernel, it is a best practice to compile kernels with only needed elements, and configure modules for most hardware.
- ☐ Modular kernels, with separate device modules, are more efficient than monolithic kernels, where device drivers are integrated into the kernel.
- ☐ If you're going to update your kernel, you should keep a copy of your current working kernel.
- ☐ There are a number of RHEL 3 kernels available for different types of CPU and levels of RAM.
- ☐ Linux kernels and related files are stored in the `/boot` directory.
- ☐ The `/proc` directory provides a window to what Linux sees in your computer.

- ❑ Changing the values of variables in the `/proc` directory can change the behavior of your running kernel. For example, setting `ip_forward = 1` enables routing.
- ❑ Kernel modules are managed by a special kernel thread, **kmod**. Additional settings can be configured through `/etc/modules.conf`.
- ❑ The **lsmod** command lists currently loaded modules; important related commands are **insmod**, **rmmmod**, and **modprobe**.
- ❑ Basic RHEL 3 modules are stored in the `/lib/modules/2.4.21-4.EL` directory.

New Kernels, the Easy Way

- ❑ Kernel version numbers are organized in *major.minor.patch* format. Red Hat adds a build number to the Linux kernels that it builds from source code.
- ❑ It's fairly easy to install a Red Hat kernel from RPM, as long as you remember to install and not upgrade. This allows you to return to the current working kernel if you have a problem.
- ❑ Sometimes, what you need is a kernel patch, which supports upgrades of one patch version number. Unfortunately, patches are not always compatible with Red Hat built kernels.
- ❑ It's important to create a new boot floppy whenever you install a new kernel.
- ❑ When you install a Red Hat kernel from RPM, the process should automatically update your boot loader (GRUB or LILO).

Kernel Sources

- ❑ The kernel source tree is accessible through `/usr/src/linux-2.4`, which is normally linked to the actual directory with kernel source files.
- ❑ Kernel sources can be loaded from the kernel-source RPM or from a Linux kernel tarball downloaded from a site such as ftp.kernel.org.

Recompiling a Kernel

- ❑ Your current kernel configuration is stored in the `config-versionnumber` file in the `/boot` directory.
- ❑ You can modify kernel settings from the `/usr/src/linux-2.4` directory with one of the following commands: **make config**, **make menuconfig**, or **make xconfig**. The last two lead to a long series of menus.

- ❑ Once you've made the proper backups and boot disks, set the `EXTRAVERSION` variable in your Makefile, run the **make mrproper** command, and placed current settings in `/usr/src/linux-2.4/.config`, you're ready to modify your kernel.
- ❑ Once you've settled on and saved your changes, run the **make dep**, **make clean**, **make bzImage**, **make modules**, **make modules_install**, and **make install** commands, you've compiled your new kernel, and it should be ready for use from the GRUB boot loader.

The cron and at Systems

- ❑ The cron system allows any user to schedule jobs so they run at given intervals.
- ❑ The at system allows users to configure jobs to run once at a scheduled time.
- ❑ The **crontab** command is used to work with cron files. Use **crontab -e** to edit, **crontab -l** to list, or **crontab -d** to delete cron files.
- ❑ The `/etc/cron.allow` and `/etc/cron.deny` files are used to control access to the cron job scheduler.

SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully, as there may be more than one correct answer. Choose all correct answers for each question. This is just to test your understanding of the material; Red Hat no longer includes multiple choice questions on the RHCE or RHCT exams.

Shell Configuration Files

1. The system-wide file associated with the bash shell is:
 - A. /etc/shells.conf
 - B. /etc/startup.sh
 - C. /etc/profile
 - D. There is no system-wide shell startup file; all such files reside in individual user directories.

Setting Up and Managing Disk Quotas

2. You are running an ISP service and provide space for users' Web pages. You want them to use no more than 40MB of space, but you will allow up to 50MB until they can clean up their stuff. How could you use quotas to enforce this policy?
 - A. Enable grace periods; set the hard limit to 40MB and the soft limit to 50MB.
 - B. Enable grace periods; set the soft limit to 50MB and the hard limit to 40MB.
 - C. Enable grace periods; set the soft limit to 40MB, and the hard limit to 50MB.
 - D. None of the above.
3. If you wanted to configure quotas for everyone on the /home directory, what would you add to the options in /etc/fstab?
 - A. quota
 - B. usrquota
 - C. grpquota
 - D. userquota

The Basics of the Kernel

4. You are troubleshooting someone else's computer and are not sure what network card is inside it. You have checked the output from the **dmesg** command, but no network cards are listed,

and even though you have a bunch of compiled network modules, none are currently loaded. What could you do to load the unknown network device most quickly?

- A. Try loading each module manually.
 - B. `modprobe \*`
 - C. Nothing. The kernel will load the module when you try to connect to the network.
 - D. `modprobe -t net`
5. You can't find any modules in the `/lib/modules/kernel_version/kernel/drivers` (or similar) directories. Which of the following actions would set up these modules?
- A. Go into the `/lib/modules/kernel_version` directory and run the **make modules** command.
 - B. Go into the `/lib/modules/kernel_version` directory and run the **make modules** and **make modules_install** commands.
 - C. Go into the `/usr/src/linux-2.4` directory and run the **make modules** command.
 - D. Go into the `/usr/src/linux-2.4` directory and run the **make modules** and **make modules_install** commands.

New Kernels, the Easy Way

6. When you install an updated kernel from a Red Hat RPM, which of the following steps do you need to take?
- A. Add information on the new kernel in a GRUB stanza.
 - B. Add a new Initial RAM disk to the `/boot` directory.
 - C. Reconfigure the `/usr/src/linux-2.4/.config` file.
 - D. None of the above.

Kernel Sources

7. You're not sure which kernel source tree is active for your Linux computer. Which of the following directories should be linked to that source tree?
- A. `/usr/src/linux`
 - B. `/usr/src/linux-2.4`
 - C. `/usr/src/linux-2.4rh`
 - D. `/usr/src`

Recompiling a Kernel

8. When compiling a kernel, you've navigated to the `/usr/src/linux-2.4` directory. Which of the following would not work with the **make** command to reconfigure your kernel?
 - A. `config`
 - B. `menuconfig`
 - C. `windowconfig`
 - D. `xconfig`
9. You have just compiled a new kernel with the **make bzImage** command. You need to copy the kernel and Initial RAM disk to the `/boot` directory. What is the most efficient way to make this happen?
 - A. Copy the image from the `/usr/src/linux/arch/i386/boot` directory.
 - B. Nothing. The **make bzImage** command automatically installs the kernel to the `/boot` directory.
 - C. Install a revised Red Hat kernel RPM.
 - D. After you've set up modules, run the **make install** command.

The cron and at Systems

10. You want to schedule a maintenance job to run from your home directory on the first of every month at 4:00 A.M. Which of the following cron entries runs the specified job at the specified time?

A.	0	4	1	*	*	~/maintenance.pl
B.	4	1	*	*	*	~/maintenance.pl
C.	0	4	31	*	*	~/maintenance.pl
D.	1	4	0	0	*	~/maintenance.pl

LAB QUESTIONS

If you're pressed for time with these labs, I suggest that you run Lab 3 first. As you've read in the chapter, recompiling the kernel takes a long time. If you have a slower computer, some of the commands required to recompile the kernel take a while to complete. You can use this time to log into a different terminal and run the other labs in this chapter.

Lab 1

In this first lab, we'll look at setting up automatic connections to a shared network directory. While this lab uses files described in Chapter 4, it is focused on shell configuration files. For the purpose of this lab, assume your username is `vaclav` and you're mounting a shared NFS `/mnt/inst` directory from a remote computer with an IP address of `192.168.30.4`. You're going to mount it in `vaclav`'s home directory, in a blank directory named `inst`.

1. Select the regular user of your choice. That user should have files such as `.bashrc` and `.bash_logout`.
2. Find a shared directory on a remote computer.
3. Set up a local directory for that user as a mount point.
4. Configure commands for that user to **mount** and **umount** that remote directory. Make sure those commands run only when that user logs into his or her account.

Lab 2

In this lab, we will test the quotas created in this chapter. You'll use the basic quota settings described in this chapter, and then copy files to fill up the home directory of a user who has a quota applied. The steps required for this lab are straightforward.

1. Set up quotas on the local computer. Use the criteria described earlier in this chapter. If you don't have a separate `/home` directory partition, you can set up quotas on the top-level root directory (`/`).
2. Once you've set quotas in your `/etc/fstab` configuration file, remember to remount the partition where you've created a quota. Alternatively, you could reboot Linux, but that would take time that you may not be able to spare during either of the Red Hat exams.
3. Set up a quota for the user of your choice. Remember, when you use the **edquota** command on a specific user, you can edit the quota file directly using `vi` editor commands. Configure a hard and a soft limit for that user.
4. Log in as the user with the quota. Copy some large files to the home directory of that user.
5. Continue the copying process until you see a warning message. When you do, run the **quota** command. What do you see? Is there anything in the output that gives you warning that you've exceeded the quota?

6. Copy some additional files until you see a “Disk quota exceeded” message. Run the **quota** command again. What do you see?
7. Delete some files from that user’s home directory—at least enough to get the files under the quota limits.

Lab 3

This lab is more of a detailed kernel-building exercise than a typical lab in this book. The exercise will include concise steps on how to configure, install, and test a new kernel. While the Red Hat Exam Prep guide no longer specifies that you have to know how to recompile the kernel, it is something you will need to do at some point in time as a Linux system administrator. See the Lab Answer section at the end of this chapter for the exercise.

Lab 4

In this fourth lab, you’ll be updating your kernel from another source. One proviso—this lab will work only if there is a Red Hat RPM kernel file that is a later version from what is already installed on your computer. If you’re running RHEL 3, you can still download and use one of the Fedora development kernel RPMs for the purpose of this exercise. (While there are no guarantees, the Fedora development kernel available as of this writing works fine on my RHEL 3 computer. However, there have been reported issues with various video cards and printer configurations.)

1. Check download.fedora.redhat.com or one of the mirrors listed online at fedora.redhat.com/download/mirrors.html.
2. Download the new kernel to the `/tmp` directory.
3. Back up your current `/etc/grub.conf` configuration file, as well as the current files in your boot directory, in case something goes wrong.
4. Use the **rpm -ivh *kernelfile*** command to install the new kernel.
5. Observe the results. Check the files in `/boot`. Which files look like they’re duplicated but with a different version number?
6. Check your boot loader file. Assuming it’s GRUB, open the `/etc/grub.conf` file in a text editor. Change the *default* variable in this file to point to the new kernel. If it’s LILO, remember to run **lilo** to record the change in the MBR.
7. Reboot your computer to test the new kernel.

SELF TEST ANSWERS

Shell Configuration Files

1. ☒ C. This is the system-wide shell script executed for all users.
☒ A and B are incorrect because there are no such files. While each user does get their own hidden configuration files in their home directories, they apply only to that user. D is incorrect because answer C is correct.

Setting Up and Managing Disk Quotas

2. ☒ C. Enable grace periods; set the soft limit to 40MB, and the hard limit to 50MB. This will warn users they are over their limit after the grace period, but will make sure they do not exceed the 50MB true maximum barrier.
☒ A is incorrect because the soft limit must be less than the hard limit. B is incorrect because it is actually the same as A. D is incorrect because C does the job.
☒ B and C. The `usrquota` and `grpquota` options allow the configuration of quotas on the given filesystem on user and group bases.
☒ A and D are not valid options for `/etc/fstab`.

The Basics of the Kernel

4. ☒ D. The `modprobe -t net` command tries each module from the `/lib/modules/kernel_version/kernel/drivers/net` directory, until it reaches the end of the list or finds a module that works for a network interface.
☒ A, B, and C are incorrect. Linux needs to load the card module before you can connect to the network, so C is incorrect. You might try loading all modules manually, but that is not efficient, so A is incorrect. While the `modprobe \*` would test all options, it is also not efficient, so B is also incorrect.
5. ☒ D. These commands together compile your modules and then place them in the appropriate directory trees. Because these are kernel-related commands, they have to be run from the `/usr/src/linux-2.4` directory.
☒ A, B, and C are incorrect. The `make modules` command just compiles the modules, so A and C are not correct. The `/lib/modules/kernel_version` directory is the wrong place to apply these commands, so B is incorrect.

New Kernels, the Easy Way

6. ☒ **D**, None of the above. GRUB is automatically updated when you install a new Red Hat kernel from a Red Hat RPM; however, you may want to change the value of *default* in */etc/grub.conf*. A new Initial RAM disk file, *initrd-versionnumber* is also added to the */boot* directory. You don't need to change */usr/src/linux-2.4/.config* file unless you want to recompile this new kernel.
- ☒ **A**, **B**, and **C** are all not required and are therefore incorrect.

Kernel Sources

7. ☒ **B**. The */usr/src/linux-2.4* directory is normally soft-linked to the actual directory with the active kernel source files.
- ☒ **A**, **C**, and **D** all specify directories that aren't linked to the Linux kernel source tree. (However, the situation was different before Linux kernel version 2.4; answer **A** was correct for older Linux distributions, such as Red Hat Linux 7.0.)

Recompiling a Kernel

8. ☒ **C** is correct because there is no kernel configuration utility based on the **make windowconfig** command.
- ☒ **A**, **B**, and **D**; when in the */usr/src/linux-2.4* directory, the **make config**, **make menuconfig**, and **make xconfig** commands all call kernel configuration utilities.
9. ☒ **D**. The **make install** command in RHEL 3 automatically loads the files and data related to your recompiled kernel in the */boot* directory. It also installs a new Initial RAM disk and revises the GRUB boot loader.
- ☒ **A**, **B**, and **C** are all incorrect. While the recompiled kernel is stored in the directory noted in answer **A**, this action does not create an Initial RAM disk. **B** is incorrect as more action is required. **C** is incorrect as the revised Red Hat kernel RPM probably does not reflect your custom kernel configuration.

The cron and at Systems

10. ☒ **A**. The syntax for cron is minute, hour, day of month, month of year, weekday, and then the command. This answer corresponds to 4:00 A.M. on the first of every month.
- ☒ **B** executes at 4 minutes after 1 in the morning for every day. However, there are only four time fields, not five, so the entire line would be considered invalid. **C** is incorrect because it runs the job at 4:00 A.M. on the 31st of the month, and then only if the month has 31 days. **D** is incorrect because it executes the program at one minute after 4:00 A.M. And there are only four time values in this line, which makes it invalid.

LAB ANSWERS

Lab 1

This lab has two purposes: it is designed to help you understand mounted network directories and the login process. You can substitute the user, the shared network directory, and directories of your choice. But based on the premises in this lab, I would take the following steps:

1. Log in as user `vaclav`. Create the specified directory. For this lab, you would use the **mkdir** / **home/vaclav/inst** command.
2. Test the network connection. Mount the remote NFS directory on the directory that you just created. For this lab, use the following command:

```
# mount -t nfs 192.168.30.4:/mnt/inst /home/vaclav/inst
```

3. Run the **mount** command by itself. If you've successfully mounted to the shared directory, you should see it at the end of the list of mounted directories.
4. Unmount the network connection. For this lab, you would use the following command:

```
# umount /home/vaclav/inst
```

5. Add the commands specified from steps 2 and 4 to the local `.bashrc` and `.bash_logout` configuration files. Remember, since these files start with a dot, they are hidden.
6. Test the result. Log out, and log back in. Check your mounted directories. If the command in `.bash_logout` does not work, you'll probably see the shared directory mounted multiple times.

Lab 2

The purpose of this lab is to get you some more practice with creating quotas for users. It's quite possible that you'll have to configure quotas on the Red Hat exams. While you may not have to test quotas in the way described in this lab, it will help you become familiar with the error messages that you'll see when you exceed a hard and then a soft quota limit.

Lab 3

Before we can build a new kernel, we have to ensure we have all the correct RPM packages. You could do so by checking a list of RPMs as described. Alternatively, you can start the Package Management utility with the **redhat-config-packages** command. From this GUI utility, make sure you have the Kernel Development package group installed. As with the rest of this chapter and the

Red Hat exams, this assumes that you have a PC with a 32-bit Intel type CPU. The procedures for other CPUs vary, and are not as of this writing covered on the Red Hat exams.

The following list of RPMs are associated with the source code.

```
kernel-source-*
glibc-kernelheaders-*
glibc-devel-*
cpp-*
ncurses-*
ncurses-devel-*
binutils-*
gcc-*
```

1. When you install RHEL 3, you've probably already installed most of these packages. Alternatively, it may be faster to install the Kernel Development package group using the Package Management utility. This utility automatically takes care of any dependencies.
2. Navigate to the `/usr/src` directory with the `cd /usr/src` command. Run the `ls -l` command. You should see a link between the `linux-2.4` directory and the location of your source code files. In RHEL, that is by default the `/usr/src/linux-2.4.21-4.EL` directory.
3. Navigate to the `/usr/src/linux-2.4` directory. You'll be running the remaining kernel configuration commands from this directory.
4. Set up a unique name for the kernel that you're about to modify. Open the Makefile file in a text editor. Look for the `EXTRAVERSION` variable. Red Hat adds this variable as a suffix to the recompiled kernel. Modify this variable as desired; save and exit from Makefile.
5. Jot down the value of the `EXTRAVERSION` variable here: _____
6. Determine the correct CPU on your hardware. Use the command

```
# cat /proc/cpuinfo
```

7. Jot down the CPU model name here: _____
8. Run the `ls /usr/src/linux-2.4/configs` command. You'll see a list of available default kernel configuration files. Find the file associated with your CPU. If your computer has more than one CPU, use the `smp` version of the kernel, if available. If your computer has more than 4 GB of RAM, use the `hugemem` version of the kernel, if available. Save it in the `/usr/src/linux-2.4/.config` file.
9. Make sure you're in the `/usr/src/linux-2.4` directory. Clean up any stray source code from previous kernel reconfigurations with the following command.

```
# make mrproper
```

10. Wait until the messages are complete. Problems are rare at this stage.
11. Next, it is time to configure your kernel, using one of the three major tools:
 - **make config** A line-by-line tool that gives you a choice with all kernel options
 - **make menuconfig** A text-based menu that allows you to select just the changes you want
 - **make xconfig** A GUI interface that works only in the X Window System
12. Set the processor type to match your hardware (for example, Pentium, Pentium II, Pentium III, Pentium IV).
13. Return to the kernel configuration tool of your choice. Turn off all unneeded devices. Some possible unneeded devices are in the following categories:
 - ISDN Subsystem
 - I2O
 - Old CD-ROMs
 - Amateur Radio
 - Telephony Support
 - Symmetric Multiprocessing Support
 - MTR Memory Support
14. Be sure to turn on Kernel Loadable Modules support.
15. Save your changes and exit.
16. When you save the new configuration, the kernel configuration tool overwrites your `/usr/src/linux-2.4/.config` file.
17. Resolve all kernel dependencies (between sources) with the following command. This will produce a lot of output and may take several minutes.

```
# make dep
```
18. Prepare the source code directories to create the new kernel with the following command:

```
# make clean
```
19. Once your dependencies are resolved, it's time to build a new compressed kernel image, with the following command:

```
# make bzImage
```
20. This is the actual kernel build, which will take some time. You may take this opportunity to log into another terminal and run one of the other labs.

74 Chapter 5: Kernel, cron, and User Administration

21. The easiest way to see if the kernel build worked is to run the following command immediately after the messages from **make bzImage** command stop:

```
# echo $?  
0
```

22. If you got a 0, everything worked (success). Any other result indicates a failure during the kernel build process. In that case, go back and reconfigure your kernel to make a configuration that works.

23. Check for the existence of two new files. Run this command:

```
# ls -l System.map arch/i386/boot/bzImage
```

It should show you two files, a relatively small System.map and a much larger bzImage.

24. Make the loadable modules that will be used by this kernel.

```
# make modules
```

25. Install the new custom kernel files into their correct locations and update your boot loader so that it knows about your new kernel. The **make install** command should perform all of these tasks.

26. Check to see that the **make install** command worked. Based on the EXTRAVERSION variable that you set earlier, check your /boot directory. You should see at least a new initrd, System.map, and vmlinuz file in this directory, with this variable as a suffix. Otherwise you'll need to copy these files yourself. Also, check your boot loader configuration file (/etc/grub.conf for the default GRUB boot loader).

27. If the **make install** command didn't put an initial RAM disk (initrd) into the /boot directory, you'll have to create one with the following command (if your version and EXTRAVERSION variables are different, revise this command accordingly):

```
# mkinitrd /boot/initrd-2.4.21-4.ELcustom1 2.4.21-4.ELcustom1
```

28. Congratulations, you have just installed a custom kernel on your new system. As long as you also have your original kernel in your boot loader menu, test it out!

29. Run the **reboot** command. You should see both kernels in the boot loader menu. Try your custom kernel!

Lab 4

Assuming everything works with the updated Red Hat RPM kernel package, you should not have to update anything, especially if your boot loader is GRUB. The steps described in the lab should help you confirm this through the appropriate configuration files on your RHEL 3 computer.