CHAPTER – 5

USER MANAGEMENT

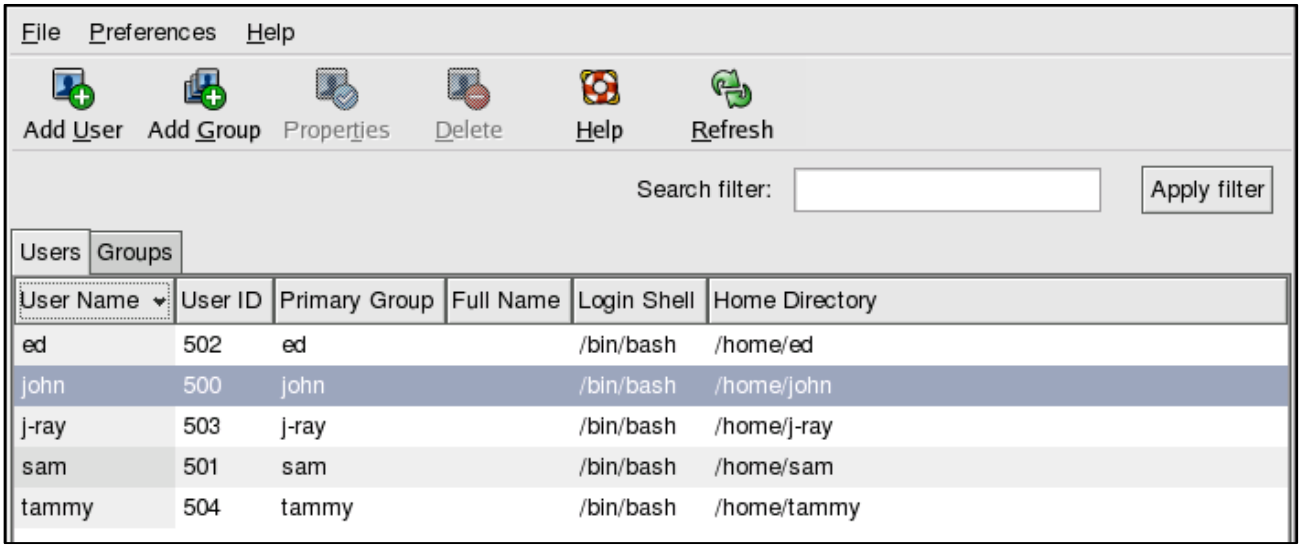
CREATING USER ACCOUNT:

When we first started our Linux system after installation, we were given the opportunity to create one or more user accounts using the **Setup Agent**. If we did not create at least one account (not including the root account) we should do so now. We should avoid working in the root account for daily tasks.

There are two ways to create new and/or additional user accounts: using the graphical **User Manager** application or from a shell prompt.

To Create A User Account Graphically Using The User Manager:

1. Select **Applications** (the main menu on the panel) => **System Settings** => **Users & Groups** from the panel.
2. If we are not logged in as root, we will be prompted for our root password.
3. The window shown below will appear. Click **Add User**.



4. In the **Create New User** dialog box, enter a username (this can be an abbreviation or nickname), the full name of the user for whom this account is being created, and a password (which we will enter a second time for verification). The name of this user's home directory and the name of the login shell should appear by default. For most users, we can accept the defaults for the other configuration options.
5. Click **OK**. The new user will appear in the user list, signaling that the user account creation is complete.

To Create A User Account From A Shell Prompt:

1. Open a shell prompt.
2. If we are not logged in as root, type the command su - and enter the root password.

3. Type useradd followed by a space and the username for the new account we are creating at the command line (for example, useradd *kundan*). Press [Enter]. User account names can be anything from the user's name, initials, or birthplace to something more creative.
4. Type passwd followed by a space and the username again (for example, passwd *12345*).
5. At the New password: prompt enter a password for the new user and press [Enter].
6. At the Retype new password: prompt, enter the same password to confirm our selection.

Setting User Defaults:

The useradd command and Red Hat User Manager window both determine the default values for new accounts by reading the /etc/login.defs file. We can modify those defaults by either editing that file manually with a standard text editor or by running the useradd command with the -D option. If we choose to edit the file manually, here is what we face:

```
# *REQUIRED*
# Directory where mailboxes reside, _or_ name of file, relative to the
# home directory. If you _do_ define both, MAIL_DIR takes precedence.
# QMAIL_DIR is for Qmail
#
#QMAIL_DIR Maildir
MAIL_DIR    /var/spool/mail
#MAIL_FILE .mail

# Password aging controls:
#
# PASS_MAX_DAYS Maximum number of days a password may be used.
# PASS_MIN_DAYS Minimum number of days allowed between password changes.
# PASS_MIN_LEN  Minimum acceptable password length.
# PASS_WARN_AGE Number of days warning given before a password
# expires.
#
PASS_MAX_DAYS     99999
PASS_MIN_DAYS     0
PASS_MIN_LEN      5
PASS_WARN_AGE     7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                   500
UID_MAX                 60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                   500
GID_MAX                 60000

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
```

```
#USERDEL_CMD /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default.
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME yes
```

Blank lines and comments beginning with a pound sign (#) are ignored. All other lines contain keyword/value pairs. For example, the keyword MAIL_DIR is followed by some white space and the value /var/spool/mail. This tells useradd that the initial user e-mail mailbox is created in that directory. Following that are lines that enable us to customize the valid range of automatically assigned user ID numbers or group ID numbers. A comment section that explains that keyword's purpose precedes each keyword. Altering a default value is as simple as editing the value associated with that keyword and then saving the login.defs file.

If we want to view the defaults, we have to type the useradd command with the -D option as follows:

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

We can also use the -D option to change defaults. When run with this flag, useradd refrains from actually creating a new user account; instead, it saves any additionally supplied options as the new default values in /etc/login.defs. Not all useradd options can be used in conjunction with the -D option. We may use only the five options listed in Table below.

Options	Description
-b default_home	Set the default directory in which user home directories will be created. Replace default_home with the directory name to use. Usually this is /home.
-e default_expire_date	Set the default expiration date on which the user account is disabled. The default_expire_date value should be replaced with a date in the form MM/DD/YYYY for example, 10/15/2001.
-f default_inactive	Set the number of days after a password has expired before the account is disabled. Replace default_inactive with a number representing the number of days.
-g default_group	Set the default group that new users will be placed in. Normally useradd creates a new group with the same name and ID number as the user. Replace default_group with the group name to use.
-s default_shell	Set the default shell for new users. Normally this is /bin/sh. Replace default_shell with the full path to the shell that you want as the default for new users.

To set any of the defaults, give the -D option first; then add any of the defaults we want to set. For example, to set the default home directory location to /home/everyone and the default shell to /bin/tcsh, type the following:

```
# useradd -D -b /home/everyone -s /bin/tcsh
```

Besides setting up user defaults, an administrator can create default files that are copied to each user's home directory for use. These files can include login scripts and shell configuration files (such as .bashrc). The following sections describe some of these files.

Supplying Initial Login Scripts:

Many Red Hat Linux applications, including the command shell itself, read a configuration file at startup. It is traditional practice that these configuration files are stored in the users' home directories. In this way, each user can customize the behavior of the command shell and other applications without affecting that behavior for other users. In this way, global defaults can be assigned from /etc/profile, then those settings can be enhanced or overridden by a user's personal files.

The bash command shell, for example, looks for a file called .bashrc in the current user's home directory whenever it starts up. Similarly, the tcsh command shell looks for a file called .tcshrc in the user's home directory. You may see a repeating theme here. Startup scripts and configuration files for various applications usually begin with a dot (.) character and end in the letters rc. You can supply initial default versions of these and other configuration files by placing them in the /etc/skeldirectory. When you run the useradd command, these scripts and configuration files are copied to the new user's home directory.

Supplying an initial .bashrc file:

By supplying our users with an initial .bashrc file, we give them a starting point from which they can further customize their shell environment. Moreover, we can be sure that the file is created with the appropriate access permissions so as not to compromise system security.

The .bashrc script is run each time the user starts a new bash shell. So, security is a concern. It is also a good place to supply useful command aliases and additions to the command search path. Here's an example:

```
# .bashrc
# User specific aliases and functions
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
PATH=$PATH:/usr/bin:/usr/local/bin
export PATH
```

This sample .bashrc file creates aliases for the rm, cp, and mv commands that result in a -i option always being used (unless overridden with the -f option). This protects against the accidental deletion of files. Next, the file executes the /etc/bashrc (if it exists) to read any further global bash values. This file also sets the search path.

Supplying an initial .tcshrc file:

This following example .tcshrc file does basically the same thing as the preceding .bashrc example. However, this file (which is for the root user) has the additional task of setting the appearance of the command prompt:

```
# .tcshrc
# User specific aliases and functions
alias rm 'rm -i'
alias cp 'cp -i'
alias mv 'mv -i'
setenv PATH "$PATH:/usr/bin:/usr/local/bin"
set prompt='[%n@%m %c]# '
```

Instead of using the export command to set environment variables, the tcsh shell uses the setenv command. In the example, setenv is used to set the PATH variable. The shell prompt is set to include our user name (%n), our computer name (%m), and the name of the current directory (%c). So, if we were to use the tcsh shell as the root user on a computer named maplewith /tmp as our current directory, our prompt would appear as follows:

```
[root@maple /tmp]#
```

The .tcshrc file can also be named .cshrc. The tcsh shell is really an extended version of the csh shell (in fact, we can invoke it by the csh name). When a tcsh shell is started, it first looks for a .tcshrc file in the current user's home directory. If it can't find a file by that name, it looks for the other name, .cshrc. Thus, either name is appropriate.

Configuring System-wide shell options:

Allowing individually customizable shell startup files for each user is a very flexible and useful practice. But sometimes we need more centralized control than that. We may have an environment variable or other shell setting that we want set for every user, without exception. If we add that setting to each individual shell, the user has the ability to edit that file and remove it. Furthermore, if that setting must be changed in the future, we must change it in every single user's shell startup file.

Fortunately, there is a better way. There are default startup files that apply to all users of the computer that each command shell reads before reading the user-specific files. In the case of the bash command shell, it reads the /etc/bashrc file before doing anything else.

Similarly, the tcsh shell reads the /etc/csh.cshrc file before processing the .cshrc or .tcshrc file found in the user's home directory. The following /etc/csh.cshrc file ships with Red Hat Linux:

```
# /etc/cshrc
#
# csh configuration for all shell invocations.
# by default, we want this to get set.
# Even for non-interactive, non-login shells.
 [‘id-gn’ = ‘id –un’ –a ‘id –u’ –gt 99]
If $status then
        umask 022
else
        umask 002
```

```
endif
if($?prompt) then
if ($?tcsh) then
        set prompt = '[%n@%m %c]$'
else
        set prompt = \['id -nu'@'hostname -s'\]\$\
endif
endif
```

The /etc/cshrc and /etc/bashrc files set a variety of shell environment options. If we wish to modify or add to the shell environment supplied to every single user on the system, the /etc/bashrc or /etc/cshrc files are the place to do it.

Setting System Profiles:

Some of the most basic information assigned to each user in added from the /etc/profile file. So, if we want to change any of the following information, we can start from /etc/profile. Here are some values contained in /etc/profile:

- **PATH:**

Assigns the default PATH for the root user and for all other users. We might change this value to add paths to local directories containing applications all users need.

- **ulimit:**

Sets the maximum allowable file size the user can create from the shell to be unlimited. We can use ulimit to restrict maximum file size if we find that users are creating enormous files. As defined in the /etc/profile file, ulimit sets no limit to the size of files a user can create. However, it does prevent core files (normally created when a process crashes) from being created.

- **Environment Variables:**

Shell environment variables that are needed for standard operation are assigned in this file. These include USER (set by the id –un command), LOGNAME (same as USER), MAIL (set to /var/spool/mail/$USER), HOSTNAME (set to /bin/hostname), and HISTSIZE (which sets shell command history to 1000 items).

- **INPUTRC:**

Sets keyboard mappings for particular situations, based on the contents of the /etc/inputrc file. In particular, the inputrc file makes sure that the Linux console and various terminal windows (xterm and rxvt) all behave sanely.

The last thing that the /etc/profile file does is look at the contents of the /etc/profile.d directory and source in the files that it finds. Each file contains setting that define environment variables or aliases that affect how users can use the shell. For example the lang.sh and lang.csh files identify the locations of foreign language files. The vim files create aliases that cause vim to be used when vi is typed. The which -2.sh file defines a set of operations used by the which command. We can modify the profile.d files or add our own to have environment variables and aliases set for all of our users.

Providing Support to Users:

Creating new user accounts is just one small administrative task among many. No single chapter can adequately discuss all the tasks that are involved in the ongoing support of users. But here are few hints and procedure to ease that burden.

Creating A Technical Support Mailbox:

Email is a wonderful communication tool, especially for the overworked system administrator. People usually put more thought and effort into their email messages than into the voice messages that they leave. A text message can be edited for clarity before being sent, and important details can be cut and pasted from other sources. This makes email an excellent method for Linux users to communicate with their system administrator.

In an office with only a few users, we can probably get away with using our personal mailbox to send and receive supports emails. In a large office, however, we should create a separate mailbox reserved only for technical support issues. This has several advantages over the use of our personal mailbox:

- Supports messages will not be confused with personal, non-support-related messages.
- Multiple people can check the mailbox and share administrative responsibility without needing to read each other's personal email.
- Support email is easily redirected to another person's mailbox when we go on vacation. Our personal email continues to go to our personal mailbox.

One easy solution is to simply create a support email alias that redirects messages to an actual mailbox or list of mailboxes. For example, suppose we wish to create a support alias that redistributes email to the user accounts for support staff members Joe, Mary, and Bob. We would log in as root, edit, the /etc/aliases file, and add lines similar to the following:

```
# Technical support mailing list
Support: joe, mary, bob
```

After saving the file, we need to run the new aliases command to recompile the /etc/aliases file into a database format. Now our users can send email to the support email address, and the message he/she should use the "Reply To All" option so that the other support staff members also see the message. Otherwise, multiple people may attempt to solve the same problem, resulting in wasteful duplication of effort.

We may also choose to create a support user accout. The technical support staff would log in to this account to check messages and send replies. In this manner, all replies are stamped with the support login name and not the personal email address of a staff member.

Resetting a User's Password:

One common (if not the most common) problem that users will encounter is the inability to log in because:

- They have the Caps Lock key on.
- They have forgotten the password.
- The password has expired.

If the Caps Lock key is not on, then we probably need to reset the individual's password. We can't look up the password because Linux stores passwords in an encrypted format. Instead, use the passwd command to assign a new password to the user's account. Tell the user what the new password is (preferably in person), but then set the password to expire soon so that he/she must choose one (hopefully, a new one that is more easily remembered).

If we reset a user's password, do so with the passwd command. While logged in as root, type passwd followed by the login name we are resetting. We are prompted to enter the password twice. For example, to change the password for mary, type:

```
# passwd mary
```

After resetting the password, set it to expire so that the user is forced to change it the next time he/she logs in. We can use the change command to set an expiration period for the password and to trick the system into thinking that the password is long overdue to be changed.

```
# change –M 30 –d 0 mary
```

The –M 30 option tells the system to expire Mary's password every 30 days. The –d 0 option tricks the system into thinking that her password has not been changed since January 1, 1970.

Modifying Accounts:

Occasionally, a user needs more done to an account that just a resetting of the password. We may need to change the groups that user is in, or the drive that a home directory resides on. The following sections explain how to modify user accounts using one of two methods: usermod or Red Hat User Manager.

Modifying User Accounts with usermod:

The usermod command is similar to the useradd command and even shares some of the same options. However, instead of adding new accounts, it enables us to change various details of existing accounts. When invoking the usermod command, we must provide account details to change followed by the login name of the account. Table below lists the available options for the usermod command.

Options	Description
-c comment	Change the description field of the account. We can also use the chfn command for this. Replace comment with a name or other description of the user account, placing multiple words in quotes.
-d home_dir	Change the home directory of the account to the specified new location. If the –m option is included, copy the contents of the home directory as well. Replace home_dir with the full path to the new directory.
-e expire_date	Assign a new expiration date for the account, replacing expire_date with a date in MM/DD/YYYY format.
-f inactivity	Set the number of days after a password expires until the account is permanently disabled. Setting inactivity to 0 disables the account immediately after the password has expired. Setting it to -1 disables the option. Which is the default behavior.

-g group	Change the primary group (as listed in the /etc/group file) that the user is in. replace group with the name of the new group.
-G grouplist	Set the list of groups that user belongs to. Replace grouplist with a list of groups.
-l login_name	Change the login name of the account to the name supplied after the –l option. Replace login_name with the new name. this automatically changes the name of the home directory; use the –d and –m options for that.
-m	This option is used only in conjunction with the –d option. It causes the contents of the user's home directory to be copied to the new directory.
-o	This option is used only in conjunction with the –u option. It removes the restriction that user IDs must be unique.
-s shell	Specify a new command shell to use with this account. Replace shell with the full path to the new shell.
-u user_id	Change the user ID number for the account. Replace user_id with the new user ID number. Unless the –o option is used, the ID number must not be in use by another account.

Assume that a new employee named Jenny Barnes will be taking over Mary's job. We want to convert the mary account to a new name (-l jenny), new comment (-c "Jenny Barnes"), and home directory (-d/home/jenny). We could do that with the following command:

```
# usermod –l jenny –c "Jenny Barnes" –m –d /home/jenny mary
```

Furthermore, if after converting the account we learn that Jenny prefers the tcsh shell, we could make that change with the –s option (-s/bin/tcsh).

```
#usermod –s /bin/tcsh jenny
```

Instead, we could use the chsh command to change the shell. The following is an example:

```
#chsh –s /bin/tcsh jenny
```

The chsh command is handy because it enables a user to change his or her own shell setting. Simply leave that user name parameter off when invoking the command, and chsh assumes the currently logged in user as the account to change.

Modifying User Accounts with Red Hat User Manger:

To use the desktop to change an existing account, we can use the Red Hat User Manger. Here's how to add a new user from the Red Hat User Manger window:

1. From the Red Hat menu, click on system settings -> Users and Groups (or type redhat-config-users from a Terminal window as root user). The main Red Hat User Manager window appears.
2. Click on the user name of the account we want to modify, then select the Properties button. A user properties window appears, as shown in the picture below:

3. There are four tabs of information we can modify for the user we selected:
   - **User Data:** This tab contains the user information we created when we first added the user account.
   - **Account Info:** Click on "Enable account expiration" and type a date if we want the account to become inaccessible after a particular date. Click on "User account is locked" if we want to prevent access to the account but not delete it. (The latter is a good technique when an employee is leaving the company or if we want to look out a customer whose account is temporarily disabled. The information isn't removed, it just isn't accessible.)
   - **Password Info:** Click on "Enable password expiration" if we want to control expiration of the user's password. By default, passwords don't expire. Here are our options: "Days before change allowed" (forces the user to keep the password for at least a set number of days before it can be changed); "Days before change required" (allows the user to keep the same password for at least the number of days); "Days warning before change" (sets how many days before the password expiration day that the user is warned to change the password); "Days before account inactive" (sets the number of days after which the account is deactivated).
   - **Groups:** Select from the list of available groups to add the user to one or more of those groups.
4. Click Apply to apply the changes to the user account.

Deleting User Accounts:

Occasionally, it is necessary to remove a user account from our Red Hat Linux system. This can be done with either the userdel command or the Red Hat User Manger window.

Deleting User Accounts with userdel:

The userdel command takes a single argument, which is the login name of the account to delete. If we supply the optional –r option, it also deletes the user's home directory and all the files in it. To delete the user account with login name mary, we would type this:

```
#userdel mary
```

To wipe out her home directory along with her account, type this:

```
# userdel –r mary
```

Files owned by the deleted user but not located in the user's home directory will not be deleted. The system administrator must search for and delete those files manually. The find command comes in very handy for this type of thing. Here are a few simple examples of how to use find, to locate files belonging to a particular user, even when those files are scattered throughout a file system. We can even use the find command to delete or change the ownership of files.

Command	Description
find /-user mary	Search the entire file hierarchy (start at /) for all files and directories owned by mary and print the filenames to the screen.
find /home –user mary –exec rm –i {} \;	Search for all files and subdirectories under /home owned by mary. Run the rm command interactively to delete each file.
find /-user mary –exec chown jenny {} \;	Search for all files and subdirectories under /home that are owned by user mary and run the chown command to change each file so that it is owned by jenny instead.
find /-uid 500 –exec chown jenny {}\;	This command is basically the same as the previous example, but it uses the user ID number instead of the user name to identify the matching files. This is useful if we have deleted a user before converting her files.

There are a few common things about each invocation of the find command. The first parameter is always the directory to start the recursive search in. After that come the file attributes to match. We can use the –exec to parameter to run a command against each matching file or directory. The {} characters designate where the matching filename should be filled in when finds runs the –exec option. The \; at the end simply tells Linux where the command ends. These are only a few of find's capabilities.

Deleting User Accounts with Red Hat User Manager:

To delete a user from the Red Hat Manager window, simply click the line representing the user account, then click the delete button.

- The information about the user is removed from the /etc/passwd file; thus, the user can no longer log in.
- The home directory and all files owned by the user will still exit. However, a listing files previously owned by that user (ls -l) will show only the former user's UID, but no name, as the owner.

Tips: *We may want to transfer the ownership of the files from the old user to new user (if, for example, a new employee is taking over the work of an employee that is leaving the company). In that case, after we delete the old user, we can create a new user account using the same UID as the old account (with a new password, of course). The new user will immediately have ownership of all files owned by the deleted user.*

Checking Disk Quotas:

Limited disk space can be another source of user support calls. Red Hat Linux offers the quotas software package for limiting and displaying the amount of disk space that a user can consume. We can also use the du command to see how much disk space has been used in a particular directory (and related subdirectories). To automate the process of checking for disk space, we can create our own script. The following sections describe these ways of dealing with potential disk space problems.

Using Quota to Check Disk Usage:

A careless or greedy user can gobble up all the space on our hard disk and possibly bring our computer to a halt. By using disk quotas, we can limit the amount of disk resources, a user or a group can use up.

The quota package contains a set of tools that lets us limit the amount of disk space (based on disk blocks) and files (based on inodes) that a user can consume. Using quotas, we can limit the amount of usage (on a per-user and –group basis) for each file system on our computer. The general steps for setting disk quotas are:

- Creating quota files
- Editing the /etc/fstab file
- Creating quota rules
- Checking quotas

We set quotas on diks partitions listed in our /etc/fstab file. For computers that are shared by many users, there may be a separate /home or /var partition where users are expected to put all their data. That kind of partition boundary can prevent an entire disk from being consumed by one user. Quotas on the /home or /var partition can make sure that the space within those partitions are shared fairly among our computer's users.

The procedure that spans the next few sections assumes that we have a separate /home partition on our computer for which we want to create quotas. We could use any partition, not just the /home partition shown in the procedure. For example, if we have only one partition mounted at the root of the files system (/), we could set quotas for our entire file system by replacing /home with / in the following example:

Editing the /etc/fstab file:

We need to add quota support to the file system. To do that, edit the /etc/fstab file and add the usrquota option to field number four of the partition for which we want to set quotas. Here is an example of a line from /etc/fstab:

```
/dev/hda21     /home          ext3    defaults, usrquota, grpquota 1 2
```

Here, the /home file system is used to allow disk quotas for all usere's home directories under the /home directory.

Before the usrquota option can take effect, the file system must be remounted. This happens automatically when we reboot, which we will have to do if we are getting quotas for the root (/) file system. Otherwise, we might be able to user the umount and mount commands to cause the usrquota option to take effect.

Creating quota files:

We need to have aquota.user and/or aquota.group files in the root directory of the partition on which we want to establish disk quotas. To add quotas based on individuals users, we need an aquota.user file, while aquota.group is needed to set quotas based on groups. One way to create these files is with the quotacheck command. Here is an example of the quotacheck command to create an initial aquota.user file:

```
#quotacheck –c /home
```

A /home/aquota.user file is created from the previous command. (To create an initial aquota.group file, type touch /home/aquota.group). Next, we must create the disk usage table or the partition. Here's an example of how to do that:

```
#quotacheck –vug /home
```

The quotacheck command in this example looks at the file system partition mounted on /home and builds a table of disk usage. The –v option produces verbose output from the command, the –u option causes user quotas to be examined, and the –g option causes group quotas to be examined. Permissions on the tow files are set so only root can access them (chmod 600 aquota.*).

Creating a quota startup script:

If the quota package doesn't include a startup script (and it doesn't with the current Red Hat Linux), we can create our own. We want this script to check quotas (quotacheck command), start the quota service (quotastart command) and turn off the services (quotaoff command).

Open a new file called /etc/init.d/quota as root user, using any text editor. Here is an example of the content we can add to that file:

```
#! /bin/bash
# init file for quota
#
#description: checks disk quotas
#
# processname: quota
#chkconfig: -90 90
# source function library ./etc/rc.d/init.d/functions
  Case "$1" in
Start)
Echo –n "checking quotas:"
Daemon /sbin/quotaheck –avug
Echo
Echo –n "Starting quotas:"
Daemon /sbin/quotaon –avug
Echo
;;
Stop)
Echo –n "shutting down quotas:"
Daemon /sbin/quotaoff – a
Echo
;;
Restart)
```

```
$0 stop
$0 start
;;
*)
Echo “Usage: quota {start | stop | restart}”
Exit 1
Esac
Exit 0
```

The quota script, when started, first runs the quotacheck command to check all file systems form which quota checking is on. Then it turns on quota checking with the quotaon command.

Turn on the quota, as described in the previous step, we need to executable and set it to start qutomatically when we start red hat Linux. To do those things, type that following as root user:

```
#chmod 755 /etc/init.d/quota
#checkonfig on quota
```

At this point, links are created so that our quota script starts when Red Hat Linux boots.

Creating quota rules:

We can use the edquote command to create quota rules for a particular user or group. (Valid users and groups are listed in the /etc/paswwrd and etc/group files, respectively). Here is an example of an edquota command to set quotes for a user named jake.

Note: The edquota command uses the vi next editor to edit our quota files. To use a different editor, change the value of the EDITOR or VISUAL environment varable before running edquota. For example, to use the emacs editor, type the following before running edquota:

```
# export EDITOR = emacs
# edquota –u jake
```

```
Disk quotas for user jake (uid 501)
  Filesystem             blocks    soft    hard    inodes    soft  hard
  /dev/hda2                 596       0       0         1       0     0
~
~
~
"/tmp//EdP.aBY1zYC" 3L, 215C
```

This example shows that user quotas can be set for the user jake on the /dev/hda2 partition (which is /home in our example). Currently, jake has used 596 blocks (a block equals 1K on this ext3 file system). One file was created by jake (represented by 1 inode). To change the disk usage limits, we can edit the zeros (unlimited use) under the soft and hard heading for blocks and inodes.

Soft limits set limits that we don’t want a user or group to exceed. Hard limits set the boundaries that we will not let a user or group exceed. After a set grace period that a soft limit is exceeded (which is seven days, by default), the soft limit becomes a hard limit. (Type **edquota –t** to check and change the grace periods that we have set).

Here is an example of how the line in the previous edquota example could be changed:

```
/dev/hda2      596     512000  716800  1       800     1000
```

In this example, the soft limit on the number of blocks that the user jake could consume on the /dev/hda2 device (/home) is 512000 blocks (or 500 MB) the hard limit is 716800 blocks (or 700 MB). Soft and hard limits on inodes are 800 and 1000, respectively. If either of the soft limits are exceed by the user jake, he has seven days to get back under the limit, or he will be blocked from using any more disk space or inodes.

Further attempts to write to a partition after the hard limit has been exceeded results in a failure to write to the disk. When this happens, the user that tries to create the file that exceeds his limit will seet the message like the following:

```
Ide0 (3, 2): write failed, user block limit reached.
cp: writing 'abc.doc': Disk quota exceeded
```

Instead of assigning quotas to users, we can assign quotas to any group listed in the /etc/group file. Instead of the –u option to edquota, use the –g options followed by a group name.

Updating quota settings:

After we have changed quota settings for a user, we should re-run the quotacheck command. We should also run the quotacheck command periodically, to keep the quota records up to date. One way to do that is to run the quotacheck command weekly using a cron entry.

Checking quotas:

To report on how much disk space and how many inodes each user on our computer (for which we have set quotas) has consumed, use the repquota command. Here is an example of the repquota command for reading quota data relating to all partitions that are using quotas:

```
# repquota –a
*** Report for user quotas on device /dev/hda2
Block grace time: 7days: Inode grace time: 7days
                        Block limits                File limits
User            used    soft     hard  grace     used  soft  hard  grace
root  --   1973984       0        0               2506     0     0
jake  --      1296     700     1700   6days          3     0     0
```

In this previous example, jake has exceeded his soft limit of 700 blocks. He currently has six days left in his grace period to remove enough files so that the soft limit does not become the hard limit.

Using du to check disk use:

We can discover the most voracious consumers of disk space using the du command. Invoke du with the –s option and give it a list of directories; it reports the total disk space used by all the files in each directory. Add an –h option to display disk space used in numbers, followed by kilobytes (k), megabytes (M), or gigabytes (G). The –c option adds a total of all requested directories at the end. The following checks disk usage for several home directories:

```
# du –h –c –s /home/tom /home/bill /home/tina /home/sally
```

This should result in a list of ally of our user's home directories preceded by the number of kilobytes that each directory structure uses. It looks something like this:

```
339M    /home/tom
81M     /home/bill
31M     /home/tina
```

```
44k     /home/sally
450M    total
```

Removing temp files automatically:

Some potential disk consumption problem are set up to take care of themselves. For example, directories for storing temporary files used by applications (such as /tmp and /var/tmp) can consume lots of disk space over time. To deal with the problem, Ret Hat Linux includes the tmpwatch facility. The /usr/sbin/tmpwatch command runs from the cron file /etc/cron.daily/tmpwatch to delete unused temp files. Here's what that file contains:

```
/usr/sbin/tmpwatch 240 /tmp
/usr/sbin/tmpwatch 720 /var/tmp
for d in /var/{cache/man,catman}/{cat?,X11R6/cat?,local/cat?}; do
     if [ -d "$d" ]; then
      /usr/sbin/tmpwatch -f 720 $d
     fi
done
```

Each day, this tmpwatch scripts runs to delete temporary files that haven't been used for a while. Files from the /tmp and /var/tmp directories are removed after 240 and 720 hours of not being accessed respectively. Temporary man page files stored in /var/cache sub-directories are also checked and deleted after 720 hour of disuse.

Sending Mail to All Users:

Occasionally, we need to send messages to all users on our system. Warning users of planned downtime for hardware upgrades is a good example. Sending e-mail to each user individually is extremely time consuming and wasteful; this is precisely the kind of task that e-mail aliases and mailing lists were invented for. Keeping a mailing list of all the users on our system can be problematic, however. If we are not diligent about keeping the mailing list current, it becomes increasingly inaccurate as we add and delete users. Also, if our system has many users, the mere size of the alias list can become unwieldy.

The following script, called mailfile, provides a simple method of working around these problems. It grabs the login names from the /etc/passwd file and sends e-mail to all users.

```
#!/bin/csh
#
#mailfile: This script mails the specified file to all users
#             of the system. It skips the first 17 accounts so
#             we do not send the email to system accounts like
#             'root'.
#
# USAGE: mailfile "Subject goes here" filename.txt
# check for a subject
#
if ('echo $1 | awk '{print $1}'' == "") then
       echo you did not supply a subject for the message.
       echo Be sure to enclose it in quotes.
       exit 1
else
       # Get the subject of the message
```

```
       Set subject = $1
endif
#
# Check for a filename
#
if  ($2 == “”) then
        echo You did not supply a file name.
        exit 2
else
        #Get the name of the file to send
        Set filename = $2
endif
        # Get the name of the file to send
        set filename = $2
endif

#
# Check that the file exists
#
if (-f $filename) then
        echo Sending file $filename
else
        echo File does not exist.
        exit 3
endif
#
# Loop through every login name, but skip the first 17 accounts
#
foreach user (‘awk –F: ‘{print $1}’/etch/passwd | tail +17’)
        #Mail the file
        echo Mailing to $user
        mail –s “$subject” $user < $filename

        #sleep for a few seconds so we don’t overload the mailer
        #on fast systems or systems with few accounts, you can
        # probably take this delay out.
        sleep 2
end
```

The script accepts two parameters. The first is the subject of the e-mail message, which is enclosed in quotes. The second is the name of the file containing the text message to send. Thus, to send an e-mail message to all users warning them about an upcoming server hardware upgrade, we can do something similar to the following:

```
mailfile “System upgrade at 5:00 pm” upgrade.txt
```

The file upgrade.txt contains the text of the message to be sent to each user. The really useful thing about this approach is that we can save this text file and easily modify and resend it the next time we upgrade the system.

Tip: *if our users log in to our system using text-based logins instead of graphical logins, we can add messages to the /etc/motd file to have them reach our users. Any text in that file will be displayed on each user’s screen after the user logs in and before the first shell prompt appears.*