1

RHEL 3 Basics

This chapter discusses some of the basics of Red Hat Enterprise Linux 3 (RHEL 3). RHEL 3 is an enterprise-class Linux distribution that provides security, stability, and may different choices in tools and applications.

The basics of using RHEL 3 includes doing such tasks as file and user management. These tasks will be crucial in the day-to-day use and maintenance of RHEL 3. Other tasks, such as listing the running processes and available resources, are important in keeping a system running once it has been set up and configured.

The topics discussed in this chapter are:

- How to use nested desktop sessions
- How to use GConf
- Virtual terminals and applications used for GUI terminals when using the desktop
  - Basic overview of the shells installed with RHEL 3 and coverage of some of the features and topics that will be used later in this chapter
  - The configuration of hardware, including detecting hardware and running utilities to use kernel modules for hardware
  - Text editors, Internet browsers, Nautilus, and the GNOME help system
  - Viewing the system logs, system performance, and disk management
  - Managing users both through the command line and with GUI utilities
  - Copying, moving, deleting, and renaming files with Nautilus and by using the command line
  - Using the GUI and terminal to monitor processes running on the server

Chapter 1

Desktop Overview

A Linux desktop environment includes a graphical server, a windowing environment, a session manager, and the applications that run within the rest of the desktop environment. The GNU Network Object Model Environment (GNOME) and K Desktop Environment (KDE) are collections of all of these things running together to generate what a user experiences as one cohesive desktop environment.

GNOME and KDE are two popular desktop environments and are based on the GIMP Toolkit (GTK) and Qt widget sets, respectively. The desktop environment that a user prefers is usually a matter of experience and taste If you were to go to a newsgroup and post the question, "Which is better, GNOME or KDE?" you'd have to don a flame-retardant suit for the heated debate that would ensue. Depending on the settings of each, GNOME and KDE applications can look wildly different from one another, because the font and color settings for the GNOME environment don't have any affect on KDE applications, and vice versa.

To simplify look-and-feel issues, RHEL 3 layers a set of themes called Bluecurve over each desktop environment, using the same fonts and colors for KDE and Gnome.

GNOME

The default desktop environment for RHEL 3 is the GNOME desktop environment. GNOME is based on the GTK tool kit, which is a GNU toolkit used for drawing the elements of the windows. GNOME is from stem to stern an entirely open source project and is distributed under the GNU license. It runs on many other platforms in addition to RHEL 3, including FreeBSD, Mac OS X, and commercial Unix software such as Solaris.

The graphical server on which the rest of the GNOME environment runs is XFree86 server. The XFree86 server is discussed in more detail in the XFree86 section later in this chapter.

Configuring GDM

GNOME Display Manager (GDM) provides a mechanism to authenticate users with a graphical login that uses the look and feel of GNOME. Depending on how GDM is configured, users can even start up different desktop environments, such as KDE. Administrators and experienced users can also configure GDM to support a variety of security policies and appearances.

To launch the configuration utility for GDM, click System Settings ➪ Login Screen (the command name is `gdmsetup`). If you are not logged into the desktop as the root user, you will be prompted for the root password before the configuration utility opens. The configuration utility opens in a window with five tabs: General, Standard Greeter, Graphical Greeter, Security, and XCDMCP.

Under the General tab, there is a section for selecting the Greeter and for setting up automatic logins. The Greeter is the screen that is displayed when prompting a user to log in. The standard greeter displays a plain login window on a colored background. The graphical greeter has images and uses different fonts, and themes can be downloaded and installed for the graphical greeter. Automatic logins and timed logins enable a user to be logged into the desktop when a server is started. However, this is not desirable in most server environments; it poses a security risk by bypassing the login mechanism.

Automatic logins allow the server to log into a session of the desktop when the computer is booted or restarted. This behavior could be convenient for terminals or kiosks where a restricted user is automatically logged in if the machine is restarted. However, automatically logging in as the root user is not a good idea, because it allows anyone with physical access to do anything as root. The automatic login will not even present the GDM screens—the desktop will log in automatically once the server is started.

Timed logins are like automatic logins because they log a user in to a desktop session upon startup or the rebooting of a server. However, the Timed Login feature pauses for the number of seconds given in the configuration before logging the user in to a desktop session.

Using Nested Sessions

A command called `gdmflexiserver` allows users to log in as other users in a new, smaller window that is nested inside the main desktop. To log in as a different user in a nested window, type:

```
# gdmflexiserver --xnest
```

This command will open a new, nested window that displays the GDM login. The `--xnest` parameter tells `gdmflexiserver` to nest the new session in the existing one. You can log into this screen as a different user to use the desktop exactly as the user would experience it. It is very important to avoid logging in as the same user who is using the main window. For instance, if you are logged in as `jdoe` in the main desktop, do not log in as `jdoe` in the nested window. The login process will allow you to do it, but you will encounter some serious issues as applications start up.

Nested login sessions can be used to debug issues that a specific user is experiencing in his or her environment. The nested session will use user's home directories and files just as it would if the login were not nested. When you are finished, use Actions ➪ Log Out to log out of the session. The nested window will then close.

Using GConf

GConf is an application that allows system administrators to set up preferences for the GNOME desktop environment without having to edit configuration files. GConf also can send signals to applications when a configuration value has been modified.

GConf allows administrators to have fine-grained control over the desktop environments by allowing preference keys to have mandatory and default values that can be defined by the administrator. When the administrator sets a mandatory value on a preference, a user cannot change the preference; however, a default value can be changed by a user.

There are so many configuration keys in the GConf repository that there is not enough room to go over them here. However, the keys can be listed by using the command-line `gconftool-2` utility with the `--recursive-list` option. For example, to open a terminal and at the command line, type:

```
# gconftool-2 --recursive-list /apps
```

This command will list all of the keys and subdirectories that are underneath the `/apps` directory. A much shorter list that contains only the first level of subdirectories can be listed by using the `--all-dirs` parameter instead of the `--recursive-list` parameter:

```
# gconftool-2 --all-dirs /apps
```

The configuration keys contain the configuration of the overall GNOME desktop and of various GNOME applications.

As an administrator, you might not want users to be able to view hidden files with Nautilus, the default file manager for GNOME. Hidden files are files and directories that either begin with a dot (.) or are listed in the directory's `.hidden` file. The configuration key that Nautilus reads is `/desktop/gnome/file_views/show_hidden_files` in order to determine whether or not to display hidden files. Setting the value of this key to true will tell Nautilus to show all files; setting this value to false tells Nautilus not to display hidden files. The configuration value can be set by using the `gconftool-2` command, but before making any changes to the GConf repository make sure that all users are logged out of GNOME and that the `gconfd-2` daemon is not running. After the `gconfd-2` daemon has been shut down, type the following at the command line:

```
# gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml
.mandatory --type bool --set /desktop/gnome/file_views/show_hidden_files false
```

The `--direct` parameter tells `gconftool-2` to access the configuration database directly, without going through the server. The `--config-source` parameter allows a configuration file to be defined, which in this case is `xml:readwrite:/etc/gconf/gconf.xml.mandatory`. The type of the value is set by `--type bool`, and finally the key is set by `--set /desktop/gnome/file_views/show_hidden_files false`, where `/desktop/gnome/file_views/show_hidden_files` is the key to be set and `false` is the value to which the key is set.

The graphical GConf repository editor is the Configuration Editor, which can be found in the Applications ➪ System Tools menu; it will remind Microsoft Windows administrators of the `regedit` utility. It should be used with the same care, since making mistakes or being careless with this tool can cause some serious issues with the GNOME desktop environment. A few additional touches that make this tool nice to use is the ability to bookmark keys so that they can easily be referenced later. Also, there is built-in documentation for the keys and preferences that describes how keys are used.

The graphical GConf editor can also be started by using the `gconf-editor` command at the command line.

More Information

More information about the GNOME desktop environment can be found at `http://www.gnome.org`.

XFree86

The desktop environments use an X server to draw the components on the screen and to get input from devices such as pointers and keyboards. The X server software that comes with RHEL 3 is from the XFree86 project. The XFree86 software is network-transparent, which means that the desktop can run applications over the network without any additional software. More about running applications over a network is discussed in the "Exporting a display" section.

Configuration

If a desktop environment was selected during the installation of RHEL 3, XFree86 should already be configured and ready to go after you select the system's hardware configuration, such as the monitor, graphics card, keyboard, and mouse. Occasionally, the configuration is not completed properly or does

not function properly, which might lead to XFree86 not starting. If XFree86 cannot start, the desktop environment will be unavailable. There are four ways to configure XFree86. The first two ways are to run either the `xf86configure` or `xf86cfg` utility on the command line. The third method of configuring the server is to type `XFree86 --configure` on the command line.

Finally, the display sections of XFree86 can be configured using the `redhat-config-xfree86` command or by using the System Settings ➪ Display menu item to start `redhat-config-xfree86`. This tool allows the configuration of the display resolution, monitor settings, and video card settings. For configuring the mouse, use System Settings ➪ Mouse.

More Information

For more documentation on the XFree86 project, see `http://www.xfree86.org`.

Exporting a Display

A very useful feature of the XFree86 project is that it is a network-based X server. That means that any GUI application using XFree86 can be displayed back to a different host, network and host security settings permitting. This is often referred to as *exporting a display*—the actual display of an application isexported from one computer to another. Every X-enabled application, when launched, looks to the `DISPLAY` variable to see if the variable is set to anything aside from `localhost` or blank. If the `DISPLAY` variable is set to the name or IP address of another server, or *host*, the application attempts to display itself on the client identified by the `DISPLAY` variable. The application is *executed* on the server on which it was invoked, but is *displayed* on the client. For information about setting and unsetting the `DISPLAY` variable, see "Bash Shell Basics" later in this chapter.

xhost

The `xhost` command can be used to grant or revoke permissions for remote servers (systems) to display X applications on the client on which `xhost` is run. The `xhost` command adds and subtracts hosts to and from an access list that controls which remote hosts can generate a display on the local client. The access list prevents unauthorized access so that remote hosts cannot display applications on a local client.

The `xhost` command can take a `+` as a parameter without specifying an IP address, which tells the X server to accept connections from any host. Unless you have really good reasons for doing this—being lazy not among them—making your X client so promiscuous is not a good idea.

To run an application on a server named `remote server` but display it on a server named `local.server`, set up the access list on `local.server` by typing as a manual user:

```
# xhost +remote.server
```

Secure Shell (ssh)

Because of security concerns, `ssh` is installed rather than Telnet by default in RHEL 3. The `ssh` client also supports exporting displays, with the added bonus of sending the data over an encrypted connection. The `ssh` client forwards the application display when the `-X` parameter is used with `ssh` to connect to a remote server.

Troubleshooting

Even if a client has been configured to allow connections using `xhost`, other settings may prevent it from being able to display remote applications. A client must be listening on a TCP port (by default port 6000) to display an exported application, and that port must not be blocked by a firewall.

The `/etc/X11/gdm/gdm.conf` file is a configuration file that tells `X` how to start. The `command` variable in the file is set to the command that is used to start the `X` server. When the `X` server is started, one of the optional commands is `--nolisten tcp`, which tells `X` to not listen on a TCP port. The client's `command` parameter should look something like this:

```
command=/usr/X11R6/bin/X
```

On a server where `X` is running and where is no need to display remote applications, it is a good idea to add `--nolisten tcp` to the line above to prevent the server from exposing port 6000. Any port that is exposed becomes a potential opportunity for exploitation, even if the possibility is fairly small. To stop `X` from listening on a TCP port, add `--nolisten tcp` to the command variable in the `/etc/X11/gdm/gdm.conf` file:

```
command=/usr/X11R6/bin/X --nolisten tcp
```

Command-Line Overview

One of the most powerful features of RHEL 3 is the command line. The command line (or terminal, or console) allows users to type in commands and run them without the need of a graphical user interface (GUI). RHEL 3 comes with many ways to use the command line. One of them is through virtual terminals, a way of having more than one console open a single display. Another is through GUI applications that emulate a terminal or console, such as gnome-terminal.

A prompt at the command line is actually a prompt from an application called a shell. A shell is responsible for accepting the input in the form of commands, modifying the command if need be, and handing it off for execution. The default shell in RHEL 3 for root is the Borne Again Shell (bash) shell, which is discussed in the following sections.

Virtual Terminals

Virtual terminals (VT) allow more than one terminal to be running, although all of the open terminals are using the same screen. Even when there is no desktop environment running, the console is actually running in a virtual terminal. To switch to a different virtual terminal while at a console, use either the Alt+Right Arrow combination or Ctr+Alt+F2. RHEL 3 comes with six virtual terminals running. This means that a user can be logged into and running a command in VT1 and also be logged into VT2 as a different user running a different command. In scenarios like these, RHEL 3 begins to show off as a true multiuser environment.

By default in RHEL 3, the desktop environment is actually running in VT7. This means that even when you are using a desktop environment the virtual terminal can be changed to show a console view. This feature provides the ability to switch to a different virtual terminal and take action in the event that the X server stops responding to mouse input, or to check on progress during installation.

gnome-terminal

The `gnome-terminal` program, which can be started with Applications ➪ System Tools ➪ Terminal, is the default GUI terminal application for the GNOME desktop. The `gnome-terminal` is a multitabbed interface to the command shell. The shell that is used in the terminal is the user's default shell. More about setting the default shell for a user can be found in the "Managing Users" section later in this chapter.

Bash Shell Basics

The default shell for the root user after an installation of RHEL 3 is the bash shell. The Bourne Again Shell (bash) is a shell that is compatible with `sh`, the Bourne shell. The bash shell interprets the commands entered by the user into the command line. For example, when a wildcard is used on the command line, it is captured by the shell. The shell expands the input by replacing the wildcards with valid input before passing it off to a command. Two such wildcards are * and ?. The * character is expanded to match none or more characters in a string. The? wildcard is expanded to match just a single character.

The shell will also expand ranges within brackets ([]). Characters inside the braces may be a range such as "a-z" or "1-4," or a list of characters. Given the files `file1`, `file2`, `file3`, and `file4` in a directory, the command `ls file[1-3]`, which is covered in more detail in the section "Managing Files and Directories," will print to the terminal or console:

```
file1 file2 file3
```

Although `ls file[1-3]` was typed into the command line, the shell interpreted the input and expanded the input to `ls file1 file2 file3`. Lists can be used as well as ranges; items in the list are not separated by any characters, such as `ls file[123]`, which is interpreted by the shell to mean the same thing as `ls file[1-3]`. The range may also be a range of alphabetical characters. Given the files `filea`, `fileb`, `filec`, `filed`, `filee`, the command `ls file[a-c]` will list:

```
filea fileb filec
```

If the name of the file could actually contain the "-" character, but you want to use it in range, just add the "-" character first—right after the first opening bracket. If the file named `file-` was also in the directory, the command `ls [-a-c]` would show the following files:

```
file- filea fileb filec
```

Another special character that is expanded by the shell is the tilde (~). The tilde, by itself, is a shortcut for "the home directory of the current user." If the tilde proceeds a valid username, it is expanded with the home directory of that user. If `jdoe's` home directory is `/home/jdoe`, `~jdoe` will be expanded to be `/home/jdoe` by the shell.

The "." (period or dot) is expanded to be the current working directory. If the current working directory (which can be found by typing `pwd` at the command prompt) is `/home/jdoe`, typing `ls` will be interpreted by the shell as if `ls /home/jdoe` had been typed in at the command line.

There may be times when it is not desirable for the shell to do interpretation on the input—when the input needs to be taken literally. The shell will not interpret anything that is inside single quotes—including variables—so single quotes can be used to specify literal input. This becomes handy if there is a

character like an "?" in the name of the file, a tilde (~), or brackets. It becomes especially useful when telling the shell not to do variable substitution.

Using Shell Variables

Variables are words made of alphanumeric characters that have a value. The bash shell supports variables that may have values assigned to them for later use. Variables have scope by user and process, so the value of the HOME variable for the root user will be different than the value of the HOME variable for any other users. Child processes started by their parent processes inherit variables from their parents, but parent processes have no knowledge of their child processes' variables.

Environment Variables

When a user logs in, certain variables are already assigned that apply to all of the processes started by the user. A common example is the PATH variable—the PATH variable includes the list of directories that are searched when a user types in a command. The value of the PATH variable can be printed to the screen by using echo at the command line, like this:

```
# echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/user/bin
```

The PATH variable contains a list of directories separated by a colon (:) as shown above. When a user types in a command, such as ls, the shell looks in /usr/kerberos/bin, then /usr/local/bin, and so on until it finds ls and executes the command. The PATH variable for each user is typically set in one of the shell's profile scripts, such as .bash_profile for the bash shell. To view the environment variables that are already assigned values, type

```
# env
```

at the command line. The env command will print a listing of the environment variables and the values assigned to them. Here is a partial, sample listing of env's output:

```
HOSTNAME=redhat1
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=192.168.0.245 44937 22
SSH_TTY=/dev/pts/0
USER=user
...
```

Variables can be set by using the following syntax at the command line:

```
# FOO=bar
```

Where FOO is the name of the variable, and bar is the value that is assigned to the variable. The value of the variable can be printed by using echo on the command line:

```
# echo $FOO
bar
```

The variable is replaced by its value, even if it is inside a string in double quotes:

```
# echo "The value of FOO is: $FOO"
The value of FOO is: bar
```

The shell will not substitute variables if they are in single quotes, because it takes them literally just as it takes other characters like wildcards literallyBy changing the double quotes to single quotes in the previous example, the output changes:

```
# echo 'The value of FOO is: $FOO'
The value of FOO is: $FOO
```

Variable Scope

Environment variables cascade down to every process that is started by a user. For example, X-enabled applications that are started by a user look to the `DISPLAY` environment variable to know where to display the application. Any environment variable can be overridden for running a single command without affecting the value of the environment variable. Also, a child process can override an environment variable for itself without affecting the value of the parent's environment variable.

When commands are enclosed inside parentheses, they are executed in a "subshell". Since a variable overridden by a child process doesn't affect a parent process, the value of a variable can be assigned without affecting the value of the parent's variable. The example below illustrates how to override the `DISPLAY` environment variable for a single command:

```
# ( DISPLAY=192.168.0.100:0.0 ; up2date )
```

The `up2date` command will use the value assigned to `DISPLAY`, `192.168.0.100:0.0`, but the main shell's environment variable `DISPLAY` remains unchanged. This comes in handy when a user uses `su` to become `root` then wants to run `up2date`. Since the user owns the variables, the root user will not have the `DISPLAY` variable set.

To illustrate the scope of environment variables between processes, a very simple shell script can be written to demonstrate variable scope (more about shell scripting can be found in Chapter 14, "Advanced System Administration")The shell script looks like this:

```
#!/bin/sh
echo $DISPLAY
```

This script, when run in a terminal, is a child process of the user's main login shell. The environment variable, `DISPLAY`, is inherited from the script's parent so it has a value without the script having to set it explicitly. As with running the command in parentheses, any value assigned to the `DISPLAY` variable will not affect the value of the `DISPLAY` environment variable.

The bash shell has a built-in function called export that allows variables to be added to the environment. But even if a variable is added to the environment in a child process, the value does not affect the parent process's variable.

Sourcing Scripts

Although the sample script in the previous example does not affect the environment variables, scripts can be called in a special way that allows them to modify environment variables. This is called *sourcing* a

script, and is accomplished by using a "." (dot or period) on the command line. If environment variables are set in a file called `myenv.sh`, the variables can be set by typing:

```
# . myenv.sh
```

Another method of sourcing scripts is to use `source`:

```
# source myenv.sh
```

Unsetting Variables

Occasionally, it is useful to undefine an environment variable. For this purpose, the `unset` command can be used, passing the name of the variable to `unset`, like this:

```
unset DISPLAY
```

After running this command, the `DISPLAY` variable will be unset. It will have no value and will not appear in the output of the `env` command.

More Information

For more information about the bash shell, type `man bash` or `info bash` at the command line. The "EXPANSION" section in the bash man pages discusses more about wildcards and other characters that are interpreted and expanded by the bash shell.

Hardware Configuration

Hardware configuration is one area where almost all distributions of Linux have grown in the last few years. RHEL 3, during installation, will detect almost all supported hardware, and support is getting better with each update. Most of the time, an administrator won't have to manually configure any hardware aside from the network cards, but sometimes there is a need for some manual configuration. To see what hardware is supported, visit `http://hardware.redhat.com/hcl/`.

The following sections explain Kudzu, the RHEL 3 hardware detection utility, listing detected hardware and adding modules if necessary, and configuring network, sound, and video cards.

Using Kudzu

Adding or changing hardware after installation of the OS has become a much simpler task with programs like kudzu. By default, Kudzu runs upon startup when the system is started into run levels 3, 4, and 5.

Listing Detected Hardware

If the drivers for hardware are not automatically installed or do not come with RHEL 3 and must be retreived from a third party, it is important to determine more information about the hardware if it's not already available. The hardware that has been detected on the system may not function correctly even though it's listed. For most hardware—sound cards, network cards, hardware PCI modems—a module must also be loaded by the kernel if support was not compiled into the kernel. For some hardware, it is

necessary to download modules from a manufacturer's site. The instructions for doing downloading, compiling, and inserting modules can often be found on the manufacturer's site and won't be discussed in detail in this book, since it varies from manufacturer to manufacturer.

Using lspci

To view the hardware on the PCI bus that has been detected in a system, type:

```
lspci
```

at the command line. The `lspci` utility formats the information in `/proc/pci` in a format that is easier to read. The output will look something like this:

```
00:00.0 Host bridge: ServerWorks CNB20LE Host Bridge (rev 06)
00:00.1 Host bridge: ServerWorks CNB20LE Host Bridge (rev 06)
00:02.0 PCI bridge: Intel Corp. 80960RM [i960RM Bridge] (rev 01)
00:02.1 RAID bus controller: Dell Computer Corporation PowerEdge Expandable
RAID Controller 2/Si (rev 01)
00:08.0 Ethernet controller: Intel Corp. 82557/8/9 [Ethernet Pro 100] (rev 08)
00:0e.0 VGA compatible controller: ATI Technologies Inc 3D Rage IIC (rev 7a)
00:0f.0 ISA bridge: ServerWorks OSB4 South Bridge (rev 50)
01:06.0 SCSI storage controller: Adaptec AIC-7880U (rev 02)
02:0e.0 Communication controller: Lucent Microelectronics Venus Modem (V90,
56KFlex)
```

This is output from on a Dell PowerEdge 2400. The first three lines contain information about the bridges. The fourth line shows information about the RAID bus controller. The following lines show information about the network card ("Ethernet controller") and video card ("VGA compatible controller").

More Information

For more information on using `lspci`, type `man lspci` or `info lspci` at the command line or use the GNOME help browser to view more documentation on `lspci`.

Using the Hardware Browser

The Hardware Browser provides a method of looking at detected hardware through the UI. To start it, click Hardware Browser on the System Tools menu. The Hardware Browser requires root permissions to run. In one side of the Hardware Browser window, the types of hardware such as CD-ROM drives, floppy disks, and network devices are listed. When the type of hardware is selected, the available devices are listed on the opposite side of the window. Detailed information about the device is listed below, including the name of the module that is used for a driver. For instance, on the same server listed in "Using lspci," the Network devices category shows the information about the network card, including the name of the driver which is `e100`.

Adding Modules

For most hardware, the RHEL 3 kernel uses modules to communicate with the hardware. These modules are analogous to "drivers" in the Microsoft Windows world. Although hardware may be detected, it may not necessarily function correctly if the correct modules are not loaded to allow the kernel to communicate with the hardware.

A couple of utilities help in listing and adding modules—`lsmod` and `modprobe`. The `lsmod` command lists the modules loaded by the kernel—basically just nicely formatted output of what is found in the `/proc/modules` file. The `lsmod` command shows if modules are being used and by what. The `modprobe` command is a way of nicely adding and removing modules to the kernel.

A partial example output of `lsmod` is:

```
e100                   56164  1
floppy                 58160  0  (autoclean)
sg                     36972  0  (autoclean)
sr_mod                 18136  0  (autoclean)
cdrom                  33696  0  (autoclean) [sr_mod]
microcode               4724  0  (autoclean)
st                     31716  0
ext3                   88104  2
jbd                    52464  2  [ext3]
aacraid                33828  3
aic7xxx               163440  0
sd_mod                 13744  6
scsi_mod              109480  6  [sg sr_mod st aacraid aic7xxx sd_mod]
```

This example is from the same system used as an example in the "Listing Detected Hardware" section. In that section, the network card (a Intel Ethernet Pro 100) was listed as using the e100 driver in the Hardware Browser. In the partial output above, notice that the e100 module is listed as being used by the kernel.

If the module for a certain piece of hardware is not loaded, it can be loaded using the `modprobe` command. Modules that are used as drivers can be found by default in the `/lib/modules/[kernel version]/drivers` directory, where `[kernel version]` is the version of the current kernel. To find out what version that is, at the command line type:

```
uname -r
```

Using the `-r` parameter will print the version of the running kernel. An example of a module that is located in the directory under the `net` subdirectory, which contains the network card drivers, is the 3c59x module (the file name is `3c59x.o`) which is used by some 3Com network cards. If a card supported by this module were installed in the system and for some reason the module had not loaded during hardware detection, it could be loaded manually into the kernel by typing:

```
modprobe 3c59x
```

If there is no device that is supported by this module, the insertion will fail. If hardware is detected that matches the module, the module will be loaded by the kernel and used to communicate with the hardware.

A device can also be unloaded by a kernel, effectively stopping support to the device. This is useful if there are conflicting devices, such as network cards—one of the cards can be disabled by unloading the kernel module. The `rmmod` command is used for this purpose. To unload the module that was loaded in the prior example, type:

```
rmmod 3c59x
```

More Information

More information about using `lsmod`, can be found by typing `man lsmod` or `info lsmod` at the command line or by using the GNOME Help Browser to view the documentation on `lsmod`. More information on `modprobe` can be found by typing `man modprobe` or `info modprobe` at the command line or by using the GNOME Help Browser to view the documentation on `modprobe`. For more information about `rmmod`, type `man rmmod` or `info rmmod` at the command line.

Configuring Network Cards

Network cards can be configured using the Network item under the System Settings menu. The Network Configuration utility has five tabs: Devices, Hardware, IPsec, DNS, and Hosts. This section will cover the Devices and Hardware tabs.

Devices

Devices are not the same as physical hardware, and it is important to keep this distinction in mind when setting up network devices. More than one device can be linked to a single piece of hardware, allowing network cards to have more than one IP address. Setting up new devices is easy under the Devices tab in the Network Configuration utility. To add a new device, click the New button. The Add New Device Type window will open with a list of the types of devices that can be added. To add a new Ethernet device, select the Ethernet controller item and click the Forward button.

In the next step, a list of hardware that is on the system will appear. Select the hardware onto which this device will be added, and click the Forward button.

In the next screen, choose between using DHCP or using a static IP address. Once the appropriate settings have been entered, click the "Forward" button. Finally, to create the device click the "Apply" button on the last screen.

Hardware

New network hardware can be configured by clicking the "New" button while the Hardware tab is active. A window will prompt you for the type of hardware—wireless, Ethernet, modem, ISDN, or token ring—and once you select the appropriate hardware type, the Network Adapters Configuration window will open. Choose the appropriate adapter from the Adapter list. The selection in the device list will automatically default to the next one in the sequence, so there is no need to change it. For almost all cards, the other values such as IRQ, MEM, IO, IO1, IO2, DMA0, and DMA1 are unnecessary. Click the "OK" button to save the new hardware.

The UI runs the `modprobe` command in the background, and if `modprobe` was successful at adding the appropriate driver module the new hardware will be ready to use once a device has been added to it.

Configuring Video

To configure video in GNOME, select the Display item from the System Settings menu or type `redhat-config-xfree86` at the command line. The Display settings window has two tabs: Display and Advanced.

Under the Display tab, the Resolution and Color Depth can be selected from lists. For these settings to take effect, you must log out of the desktop environment and log back in.

Under the Advanced tab, the model of the monitor and video card can be selected. Most modern hardware can be probed, so the settings can be detected automatically. Changing the monitor and video card also requires logging out of and back into the desktop environment so that the changes take effect.

When the configuration is modified, in either tab, the GUI writes a new version of the `/etc/X11/XF86Config` file. The previous one is saved as `/etc/X11/XF86Config.backup`, so if for some reason the changes that are made prevent the desktop from working, the backup file can be put into place, and the X server will work as it previously did, after it is restarted. The X server can be restarted by force by using the Ctrl+Alt+Backspace combination. This will stop the desktop session (very rudely) and also restart the X server, displaying the GUI Login screen after the X server has completely restarted.

See the "Virtual Terminals" section earlier in this chapter for more information about switching to a console if a previous version of the `/etc/X11/XF86Config` file needs to be recovered.

GUI Applications

The next few pages introduce GUI applications that can be used to edit files, and view documents and Web pages, and perform other tasks. One of the good things about RHEL 3 is that there are many, many applications that can be used for these purposes, but that's also a bad thing. It is easy to become confused by the number of different applications that can be used for the same thing.

Text Editors

Once in a while, it may become necessary to modify text configuration files or to view or edit other plain text files. The desktop environment comes with several GUI text editors that can be used for viewing and modifying files. The default GNOME text editor, `gedit`, supports plug-ins and the ability to open multiple documents in different tabs. Old-school Unix users will find `gvim` useful, because it is a GUI version of `vim` that supports `vi` key mappings as well as syntax highlighting and macros associated with file types.

Using gedit

To open `gedit`, click Text Editor from the Applications menu in GNOME or use the Run Application item item from the Actions menu to type `gedit` and press Enter. If the full path of a file is specified with `gedit` in the Run Application window, `gedit` will open with the specified file loaded.

Using gvim

`GVim` is a GNOME-enabled version of the `vim` text editor. It supports the same key mappings as `vim`, which are inherited from those of `vi`. It is certainly more cryptic to use for administrators and users having little experience with `vi`, but is an extremely powerful text editor when used to its full potential.

Browsing the Internet

Many programs come with the RHEL 3 desktop environment that can be used to browse the Internet. Even if a server is configured without a desktop environment, browsing the Internet is possible through a program called Lynx. The graphical Internet browsers include Konqueror and Mozilla.

Konqueror

Konqueror is an excellent browser from the KDE desktop environment. It renders pages very nicely and can be used for many other purposes aside from browsing the Web, such as viewing files, directories, and man pages. Man pages can be viewed by typing the word into the location bar, but proceeding it with a `#`. For instance, to view the man pages for `uname`, type `#uname` in the location bar and press Enter.

Mozilla

The Mozilla browser is an open source browser that is installed with RHEL 3 and is very full-featured. More information about Mozilla can be found at `http://www.mozilla.org`. Many of the other browsers that work in the GNOME desktop environment, such as Galeon or Epiphany, use the Mozilla HTML rendering engine to display HTML.

Using Nautilus

Nautilus, which can be used to browse the Internet, is a graphical desktop shell and the default file manager application that is included with GNOME. With Nautilus, files and directories (or folders) may be copied, moved, deleted, renamed, and opened.

Three methods of launching Nautilus are: using the desktop's context-sensitive menu, using the Home icon on the desktop, or selecting Home Folder from the Applications menu. To use the desktop's context-sensitive menu, right-click anywhere on the desktop (not on an application), and select Open New Window from the menu that appears. To launch Nautilus using the home icon, either click or double-click on the home icon on the desktop, or right-click on the icon and select Open from the menu. Nautilus can also be launched by selecting the Home Folder item from the Applications menu, which is usually found in the top menu bar.

Special URIs

There are special URI locations that Nautilus understands and opens. One of them, `fonts:///`, allows fonts to be installed by simply dragging and dropping them into the Nautilus window. Another special URI location, `applications:///`, allows a user to customize menus.

The URIs and their uses are listed in the following table:

URI	Description
applications:///	Contains the contents of the Application menu for the current user. Menus are modified by editing objects in this folder.
applications-all-users:///	The Application menu for all users. Changes made to this folder will appear in the menus for all users.
burn:///	If a CD-R or CD-RW drive (a CD burner) is installed, this virtual folder provides an interface for writing files to writable CDs.
fonts:///	The fonts used in the desktop environment. Fonts can be previewed by double-clicking on them in this folder. Fonts may be installed by dragging and dropping them into this folder.

Continues

URI	Description
network:///	This folder shows other servers on the same network, if there are any.
nfs:///	If the server is configured to access other NFS servers, they will appear in this folder.
preferences:///	This folder contains the preferences that are also found in the Desktop Preferences menus. The preferences in this folder are the preferences for the current user.
preferences-all-users:///	Similar to the preferences:/// URI location, this folder displays the preferences for all users.
server-settings:///	Applications that can be used to configure your server can be found in this URI location.
smb:///	Samba servers can be found in this location, as well as Microsoft Windows servers that have shares. Double-clicking on one of the servers will open a window that contains the shared folders.
start-here:///	The start-here URI location is a root folder that contains the Applications, Desktop Preferences, Server Settings, and System Settings folders.
system-settings:///	This URI location contains applications that can be used to change system settings.
themes:///	Themes for the GNOME desktop environment can be found in this folder. Themes can be installed into this folder by dragging and dropping the themes into this folder.

The Nautilus Side Pane

When Nautilus is started by any of these methods, the contents of the current user's home folder are listed in the right pane. If the side pane if visible, it will be on the left and will contain some information about the open folder. If the side pane is not visible, it can be displayed by pressing F9 or by selecting View ➪ Side Pane from the menu. The side pane in Nautilus can display information, emblems, history, notes, or a tree of folders and files. You can switch between views of the side pane by selecting one from the drop-down list at the top of the pane.

The information view shows general information about the file or folder that is open in the right pane. If the right pane shows the contents of a folder, the name of the folder is displayed in large text with the number of subitems in the folder along with the date and time the folder was last modified. The information view can be seen in Figure 1-1.

The emblem view (as shown in Figure 1-2) displays the emblems, or icons, that can be used to decorate a file or folder's icon. Adding an emblem to an icon is an easy way to add a visual notation to a file or folder. For instance, is a user wants to distinguish a file as being "cool," the "Cool" emblem can be dragged from the side panel and dropped onto the file's icon. The file's icon will change to show the emblem in the top-right corner of the icon. Nautilus will remember the file or folder's emblem and will display it when Nautilus is closed and reopened.

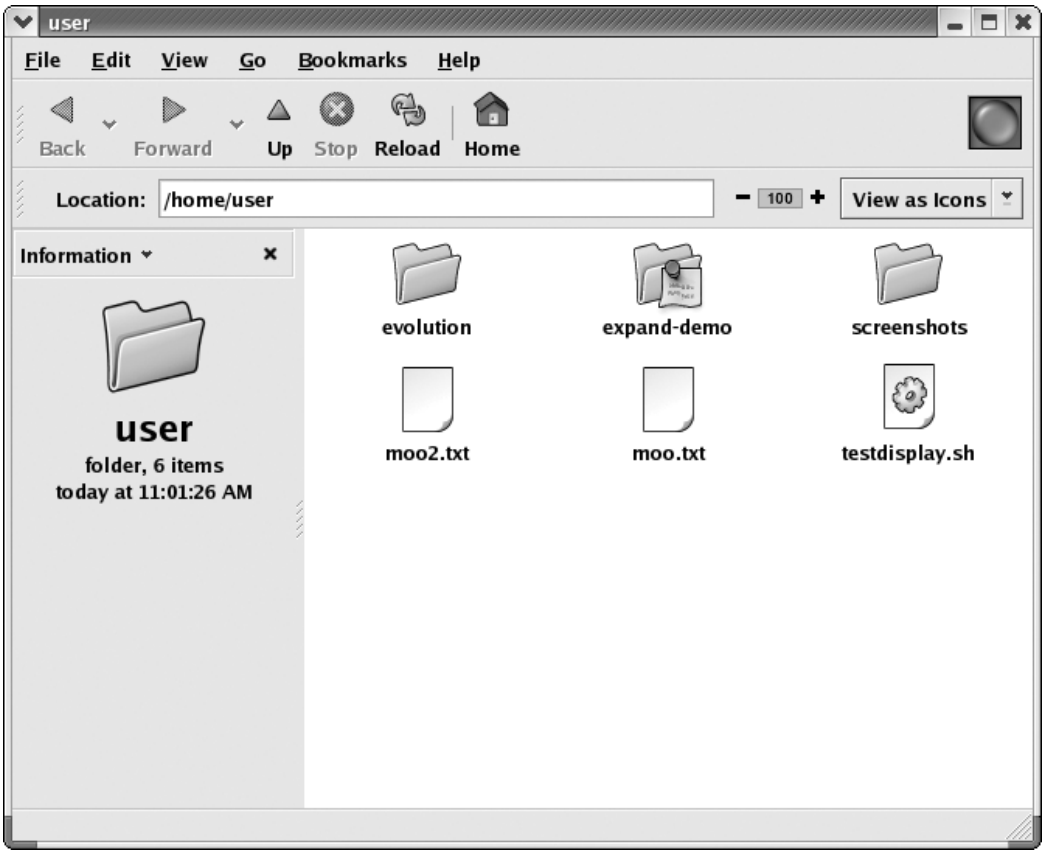

Figure 1-1

The history view (Figure 1-3) is very similar to the history view of most Internet browsers. As files and folders are opened, they are listed in the history view of the sidebar. The items in the history view can be re-opened by double-clicking on the name or icon of the item.

The notes view (Figure 1-4) displays notes that can be associated with the open folder. When notes have been added for a folder, a tiny icon shows up between the title of the view and the close button for the sidebar. The folder's icon is also altered to show that there are notes associated with the folder. These notes can by viewed either in the note view in the sidebar, or by right-clicking on the folder's icon and selecting Properties from the menu. The notes are listed in the Notes tab of the Properties window.

The sidebar's tree view will look very familiar to most users (an example can be seen in Figure 1-5) The tree view displays the folders—optionally files—in a hierarchy of folders. By default, the tree view displays only folders, but it can be configured to display files.

Using the GNOME Help Browser

The GNOME Help Browser, yelp, is a browser for viewing documentation on commands and applications. With it, info pages and man pages for commands can also be viewed.

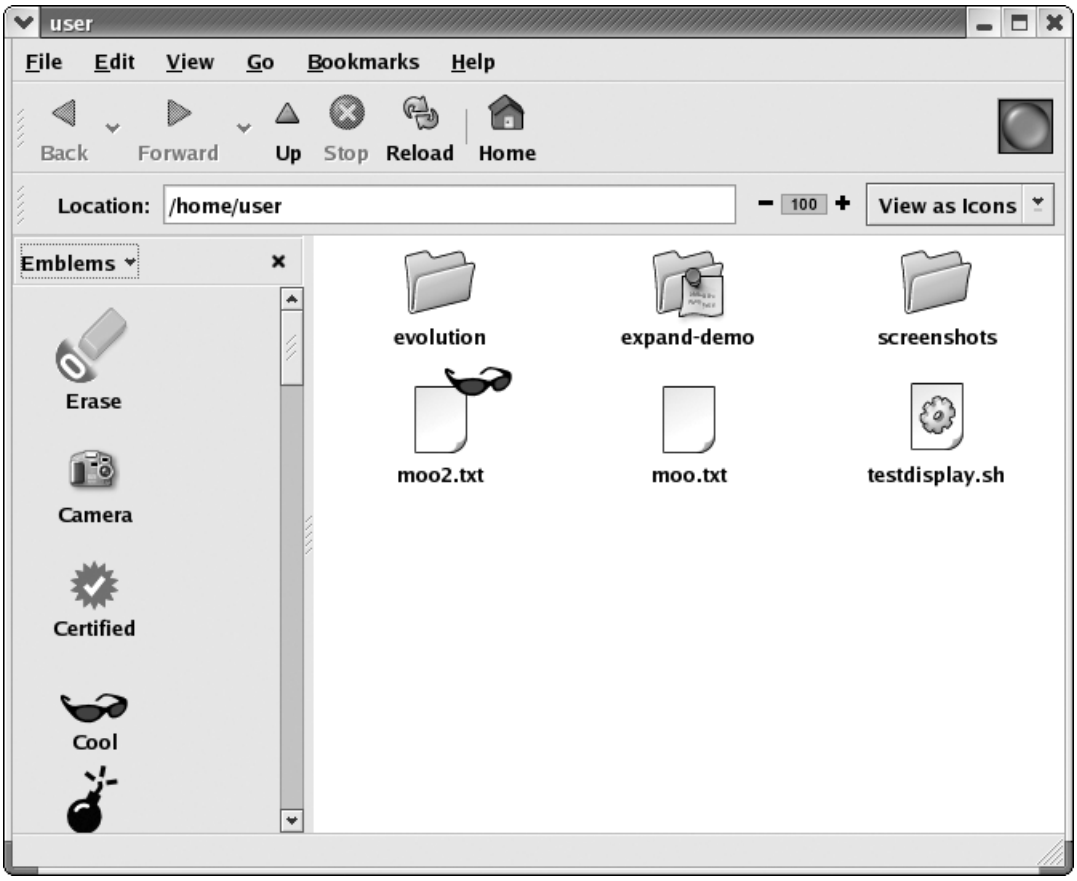

Figure 1-2

Searching for a Topic

To search for a topic in the GNOME Help Browser, select Go ➪ Index from the menu or click the button in the tool bar. As words are typed into the Search for field, the contents in the left pane will change to match what is typed into the field. The man pages of `cp` (which is described in detail in the "Managing Files and Directories" section later in this chapter) can be found by typing `cp` into the Search for field and selecting the entry in the left pane that says `cp` (1). The man page for `cp` will appear in the pane to the right.

System Administration Tools

In the following section we will cover the System Administration tools you'll be using with RHEL 3.

Viewing the System Log with the GUI

To view logs in the desktop, go to System Tools ➪ System Logs. A window like the one shown in Figure 1-6 will appear.

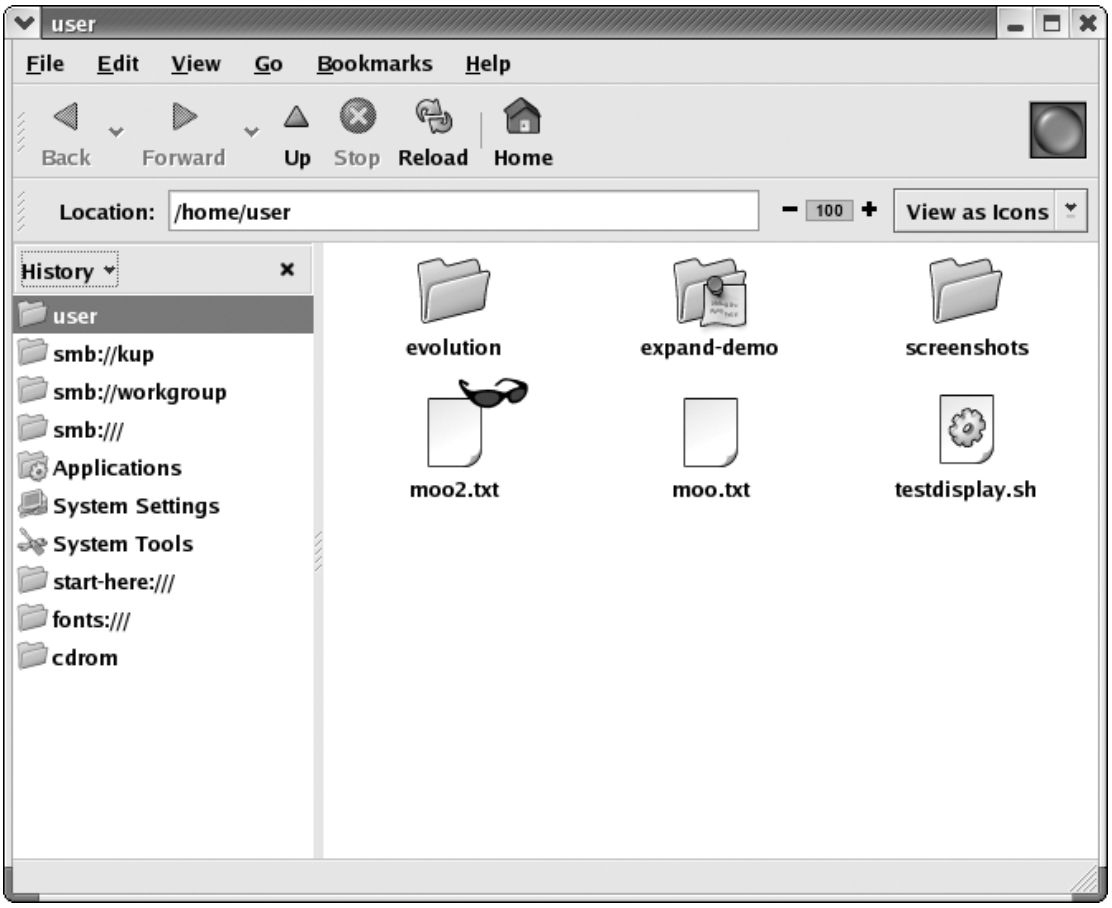

Figure 1-3

Type in a word of phrase into the Filter for box, and click the Filter button to limit the display to only entries containing that word or phrase. To show all of the entries without the filter, click the Reset button.

The System Log is one of the log entries in the list on the left side of the window. To view the system log, click the System Log line item. The contents of the log will appear in the box on the right.

Adding New Logs

The log viewer application allows new logs to be added for viewing. An example is a Web server's log, such as a site error log made by the Apache Web server for a virtual site (the default Apache access and error logs are already in the System Logs utility if Apache is installed).

To add a new log, open Preferences from the Edit menu. The Preferences window will have three tabs: Log Files, Alerts, and Warnings. New logs are added in the Log Files tab, which is shown in Figure 1-7.

Click the Add button, then type in the name, description, and location of the file system of the log, as shown in Figure 1-8. After typing in the details, click "OK" to add the log, then click the Close button on

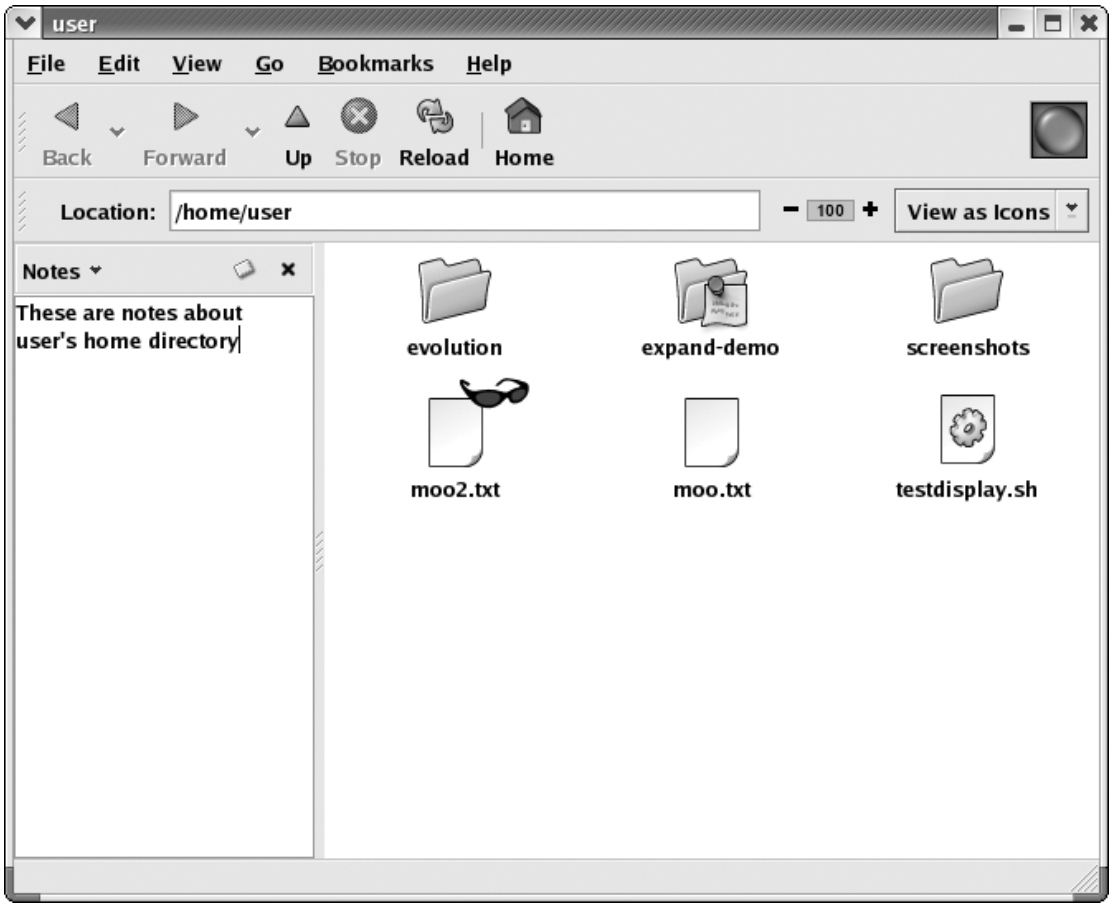

Figure 1-4

the Preferences window. The new log will now be displayed in the list and can be viewed and filtered just like the other logs.

Viewing System Logs at the Command Line

System logs can also be viewed at the command line, which is useful when gaining access to the system via SSH. Useful commands for viewing the system log include `tail`, `less`, and `grep`.

The `tail` command is used to view the last lines of a file. Many times, if there is an error in a log that is immediately apparent, it is useful to view just the last 20 or so lines of a file. To use `tail` to view the last 20 lines of the `/var/log/messages` file, type at the command line:

```
tail --n 20 /var/log/messages
```

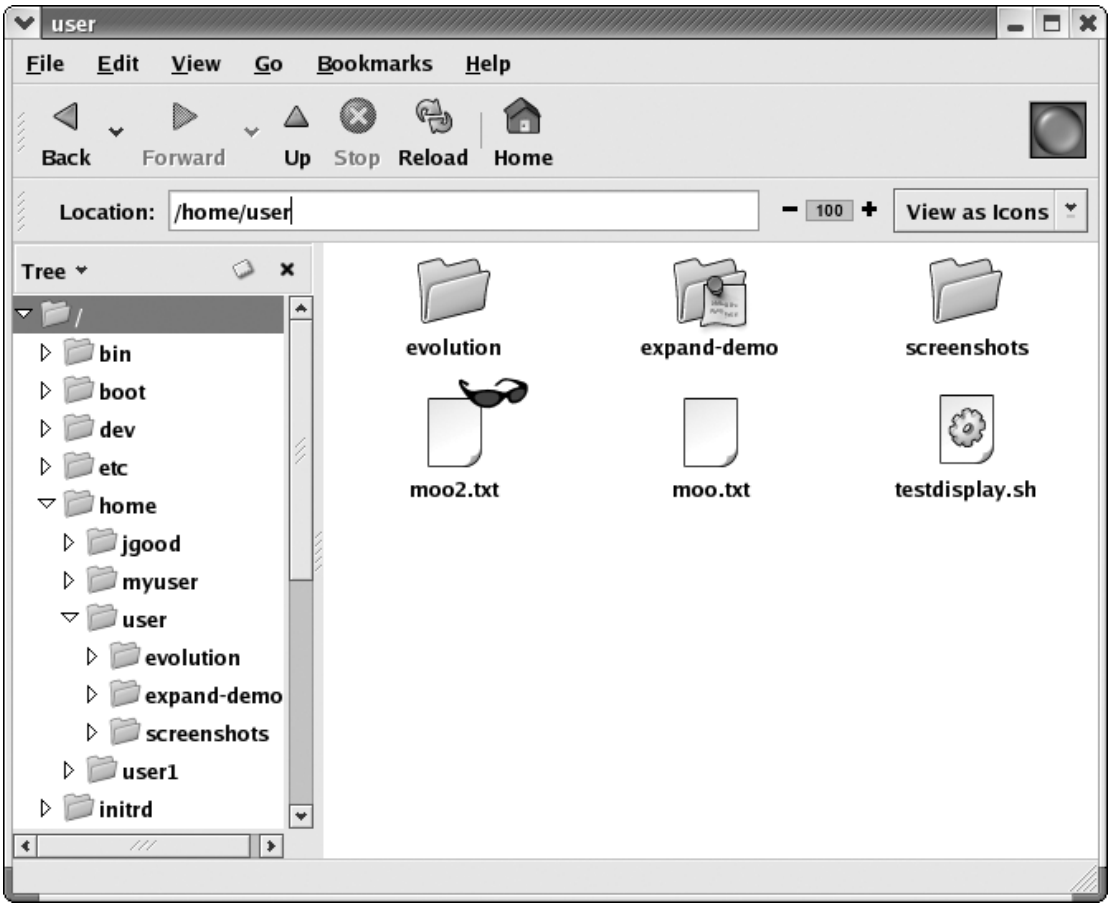

Figure 1-5

By default, `tail` will print the last 10 lines of a file. Another useful feature of tail is that when you use the `--f` parameter, `tail` will "follow" a file. Any additions to a file can be viewed as they are added. This is useful for watching a file in real time to see what is being added to it, which is very useful in for debugging.

Another command, `less`, comes in handy when viewing files. It is much better to view a file with a command like `less` than to open the file in a text editor, where it may be inadvertently modified. The `less` command doesn't read the entire file before displaying it, which can be very handy when looking at very large files. To view a file using `less`, at the command line, type:

```
less /var/log/messages
```

Use the spacebar to scroll forward through the file, Ctrl+B to go backward through the file, and `q` to quit less.

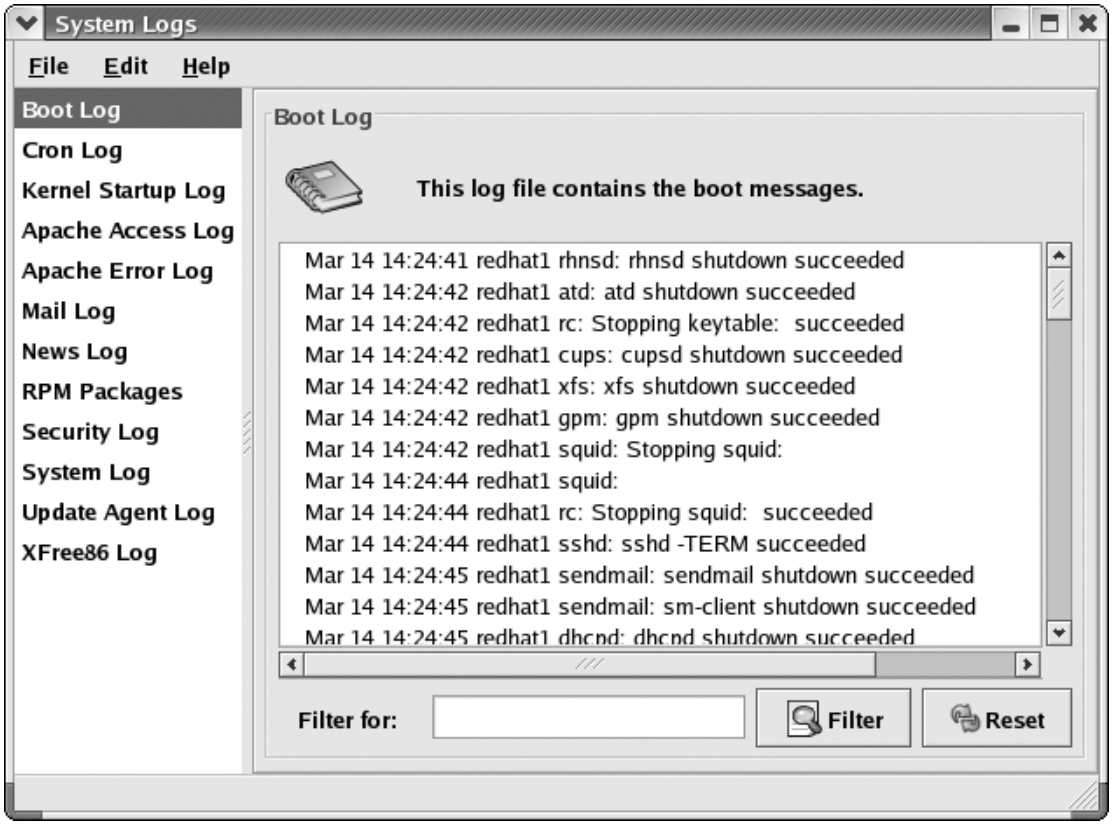

Figure 1-6

Finally, `grep` can be used to filter the lines of a file. To view all of the lines that contain the word "warning," type:

```
grep --i "warning" /var/log/messages
```

The `--i` parameter is used to tell `grep` to ignore the case of the word, so lines containing both "WARNING" and "warning" will be displayed. For more information about the usage of `grep`, type `man grep` or `info grep` at the command line.

Viewing System Performance with the GUI

The performance of the CPU, the availability of memory, and the availability of hard disk space can be viewed by using the System Monitor from the System Tools menu. The System Monitor has two tabs: Process Listing and System Monitor. The System Monitor is shown in Figure 1-9.

The System Monitor tab contains two graphs and some additional information about the available disk space by partition. The graph on the top shows the usage of the CPU. The graph on the bottom shows the usage of both the physical memory and the swap space.

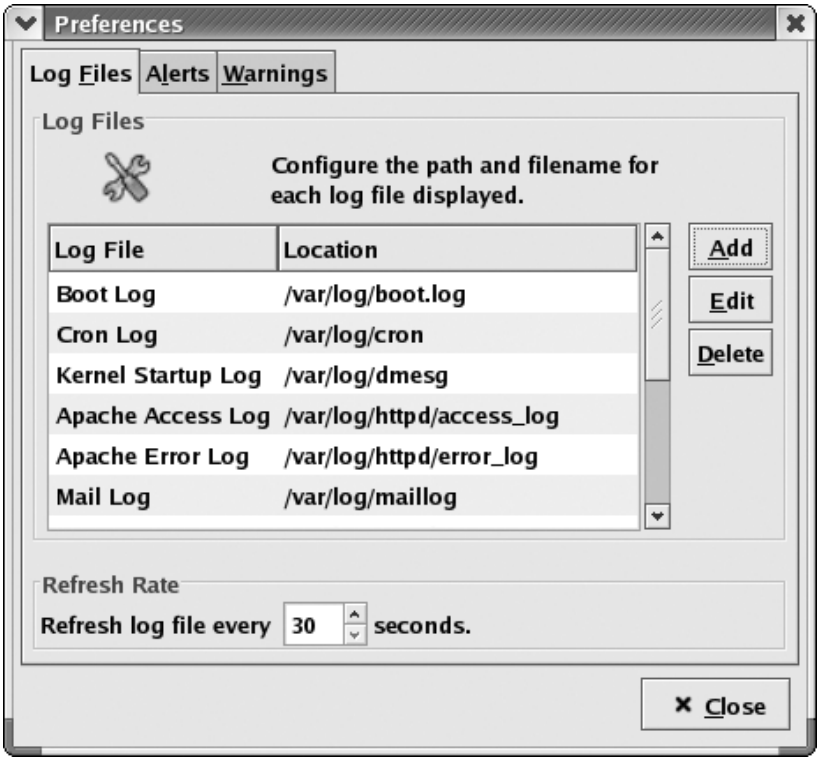

Figure 1-7

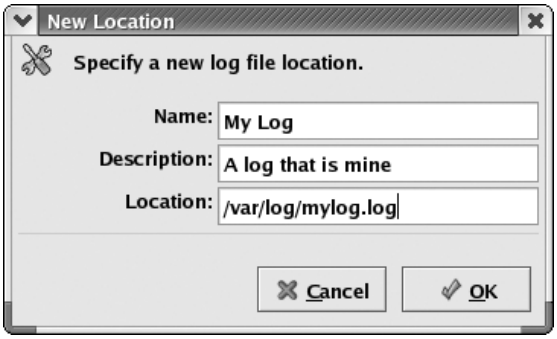

Figure 1-8

Viewing System Performance at the Command Line

The `df` command is used to view the amount of disk that is used and available. It displays the information for all of the mounted partitions. Using the `--h` parameter tells `df` to print the output in human-readable format, which means that `1000K` is rewritten as `1M` so that the output is a little more succinct and easier to read quickly.

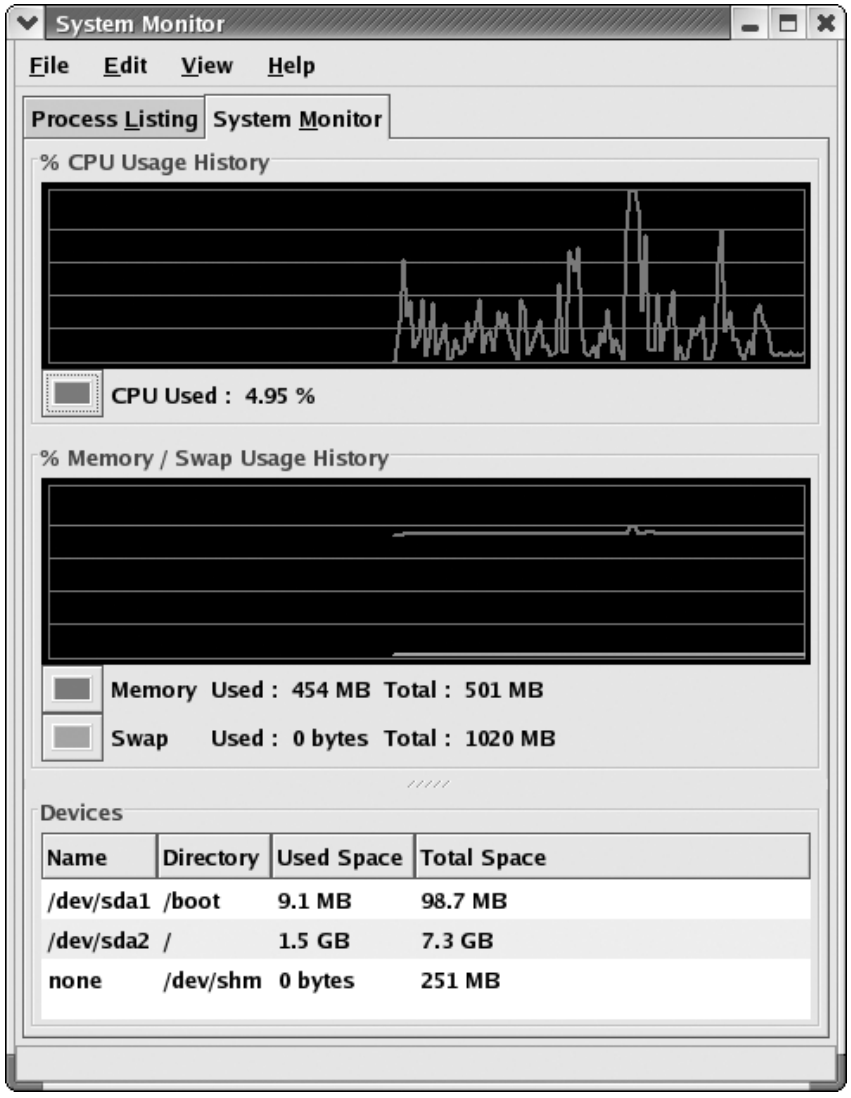

Figure 1-9

A number of commands can be used to view the amount of memory that is available on the system. One of them is `free`, which displays the total physical memory and swap space, as well as the amount of space used and available for both. Other commands that also display the amount of memory available are listed in the "Examining Processes" section.

Managing Users

In most systems, it becomes necessary to add users above and beyond those created by the system during the initial installation. In fact, at least one additional user account will usually be created during the

installation process itself, since users are strongly discouraged from running as root for day-to-day operation. Even as a system administrator, it is a good idea to run the desktop as a nonroot user, then type the root password when prompted.

Users may be managed through either GUI or command-line utilities. GUI utilities offer ease of use. Command-line utilities offer the ability to easily access a server and manage users remotely without exporting a display. Before you add too many new users to the system, it is a good idea to take a look at the default profiles and make any additions or changes.

Building Default Profiles

Default profiles are used when new users are created to give the system a template from which to copy profiles. Default profiles are also a great way to ease administration, because the time invested in finding out what works well in your environment can save time when each user is created on the server.

/etc/skel

The `/etc/skel` directory contains a number of files that are copied into a user's home directory when the user is created. When a file is changed in the `/etc/skel` directory, it will affect all users created after the file is modified. Modifying files in the `/etc/skel` directory will not affect current users. Some of the files located in the `/etc/skel` directory are `.bash_logout`, `.bash_profile`, and `.bashrc`. Any new file that is added to the `/etc/skel` directory will also be copied into the home directory of new users.

.bash_logout

The `.bash_logout` file is called by bash when the user logs out of the shell. It is common for the `clear` command to be put into the `.bash_logout` script. This is nice on the console as a security measure—someone coming up to the console cannot see who was last logged in or what commands were being run by the user.

.bash_profile

The `.bash_profile` file is one of the files copied from the `/etc/skel` directory into a user's home directory when a new user is created. The `.bash_profile` script is called by the bash shell when the user logs into an interactive login session of bash, after the shell calls the systemwide login script `/etc/profile`.

After a new installation of RHEL 3, the `.bash_profile` file looks like this:

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
unset USERNAME
```

The first action the login script takes is to source the user's `.bashrc` script if it exists (the `.bashrc` file is explained in more detail later). The second thing the script does is set the `PATH` variable for the user and export it. Exporting the variable adds it to the user's environment (see the "Using Shell Variables" section earlier in this chapter).

.bashrc

The `.bashrc` file is loaded by the shell during an interactive, nonlogin shell. The GNOME Terminal—`gnome-terminal`—is an example of an interactive, nonlogin shell unless the login option is specified (for more about the GNOME Terminal, see "gnome-terminal" earlier in this chapter. The `.bashrc` file can be used to specify aliases and functions that are used by the user—environment variables should go in the `.bash_profile` file.

Aliases can be used to tell the shell to substitute one command for another or to add default parameters to a command. For example, to alias `cp` to `cp -i` for all users (reasons for doing this are discussed in the "Copying Files and Directories with the Command Line" section with the command line later in this chapter), the line:

```
alias cp="cp -i"
```

can be added to the `.bashrc` script. This will cause the `cp` command to be run with the `-i` parameter whenever a user invokes the `cp` command.

The `umask` can also be set in the `.bashrc` file. The `umask` command sets the permissions mask on new files that are created by the user, so new files have a certain level of permissions. A `umask`, for example, can be set to prevent new files created by a user to be writable or executable by users outside the user's group. For more information about `umask`, see Chapter 15, "RHEL 3 Security."

New Files

As the system is built and more applications are added, it may be more evident that some applications have configuration files that can be included in `/etc/skel` so that users don't have to create the configuration files themselves. It may also be desirable to *prevent* users from creating configuration files themselves.

An example of a file that can be added to the `/etc/skel` directory and used by all new users is the `.vimrc` file, the configuration file for `vim`, if `vim` is installed and being used on the system. It may be desirable, for instance, to specify some options such as setting tabs to a certain width, turning automatic indenting on or off, or replacing tab characters with a number of spaces. Having common settings for applications like editors can help enforce standards.

Adding a User with the GUI

To add a user with the GUI, click the System Tools ➪ Users and Groups menu item. If you are running as a nonroot user, which you should be doing (read more about why in the "Security Basics" section in Chapter 15 "RHEL 3 Security", you will be prompted for the root password. After the password is typed in and verified, the user administration panel will appear.

To add a new user, click the Add User button in the toolbar or select File ➪ Add User from the menu. A screen will appear that looks like Figure 1-10.

Figure 1-10

The value that goes into the User Name field is the login name of the user. This login name can be changed later without affecting the user's access because the permissions of files and directories are actually tied to the UID of the user, not the login name.

The Full Name field is for the user's given name. The Password and Confirm Password fields are for the user's password. Although a user can be created with `useradd` (discussed in the next section) without specifying a password, one must be provided here before the user can be created.

The user's default shell can be selected from the Login Shell list. If the user has not specifically requested a certain shell, consider leaving it set to `/bin/bash`, which is the default shell when RHEL 3 is installed.

A user cannot log into a graphical desktop environment without a home directory, so it is a good idea to create one here. There are very few scenarios where a home directory wouldn't be created—one might be if a user is being recreated after being deleted and the user's original home directory is still on the system; another might be if the home directories will be mounted later on a network file system. The home directory's location can be specified in the Home Directory field, but it defaults to the user's login name after `/home`.

If Create a private group for this user is selected, a new group will be created with the same name as the value in User Name once the OK button has been clicked. For the user being created in Figure 1-10, the group name will be `myuser`.

If Specify user ID manually is checked, the UID field is enabled and the user's ID number can be typed in. If there isn't a very specific reason for manually specifying the user's ID, allow the system to specify the user ID. One of the reasons that we have specified the user ID is out of laziness—we had NFS working between two Linux servers, but weren't using NIS for user logins. As a consequence, the user IDs did not match between the computers, so we used this ability to manually specify an ID that existed on the NFS server.

Adding a User with useradd

The command `useradd` can be executed in a terminal to add new users and is very useful when a GUI is not available, such as when accessing the server via `telnet` or `ssh`. The `useradd` command can also be used to modify new default user information. For more information about using `useradd` to create and modify default new user information, see the "Building Default Profiles" section earlier in this chapter. To find more information about the `useradd` command than what is outlined in this section, type `man useradd` or `info useradd` at the command line, and take a look at the online help.

The `useradd` command, when run without the `-D` parameter, sets up a new user with the options passed to it on the command line. If there is no information provided—aside from the username, which is required—system defaults will be used. The home directory for the user defaults to the user's username appended to the default home directory, which is typically `/home` unless otherwise specified. However, this directory is not created unless the `-m` parameter is used with `useradd`. The following command:

```
# useradd -m jdoe
```

will create a new user with the username `jdoe` and will also create the home directory for `jdoe`, using the system's default settings. The location of the user's home directory can be overridden by using the `-d` parameter.

Setting Defaults

The `useradd` command can also be used to define default values that will be used later. If the `-D` parameter is specified with `useradd`, the command goes into maintenance mode and maintains the default information.

Default Home

The default home directory can be set by using the `-b` parameter with `useradd`. The user's login name is appended to this directory when the `-m` parameter is used with `useradd` to add a user. The `-d` parameter, when used, allows the default home directory set here to be overridden. This command:

```
# useradd -D -b /usr/local/home
```

will define `/usr/local/home` as the default base home directory.

Default Expiration Date

The expiration date is the date on which, by default, any new user account will expire. This value is set by using the `-e` parameter when the `-D` parameter is also specified. The date is specified in YYYY-MM-DD. The following command will set the default expiration date to January 1, 2005:

```
# useradd -D -e 2005-01-01
```

Default Inactive Time

The default inactive time is the number of days after the password has expired that the account will be disabled. The following command will set a new user's account to be disabled five days after the user's password expires:

```
# useradd -D -f 5
```

Default Group

A default group for new users can be defined by using the `-g` parameter. For instance, if there is a group with the name "users" that all new users should be a member of, the `useradd` command can be used to define the default group:

```
# useradd -D -g users
```

The group that is defined when this command is executed must already exist.

Default Shell

By using the `-s` parameter, the default shell can be defined. The following command will set the default shell to `/bin/bash`:

```
# useradd -D -s /bin/bash
```

The following command will work without any warnings or errors:

```
# useradd -D -s /bin/oink
```

This is fine as long as new users don't need to log into the server. If there is no valid shell, a user cannot log in even with valid user information. Many of the accounts used by daemons use `/bin/false` as a login shell.

Modifying Users with the GUI

Users can be modified in the GUI by opening the Users and Groups utility and clicking the "Properties" button once the user's record is selected in the lower pane. A multitabbed window will appear, as shown in Figure 1-11.

The user's name and other information can be modified in the User Data tab. The user's login name can be changed in the User Name field. The user's full name can be modified, and the password typed in the Password field and confirmed in the Confirm Password field.

The user's home directory can be changed in the Home Directory field, but changing the location of the user's home directory will not move the files in the old directory to the new location. The user's default shell can be modified by selecting a value from the Login Shell list.

The Account Info tab, as shown in Figure 1-12, allows a date to be set at which the user's account expires and the user's password to be locked. To set an expiration date for the user's account, check the box next to Enable account expiration and type the date into the three boxes next to Account expires (YYYY-MM-DD).

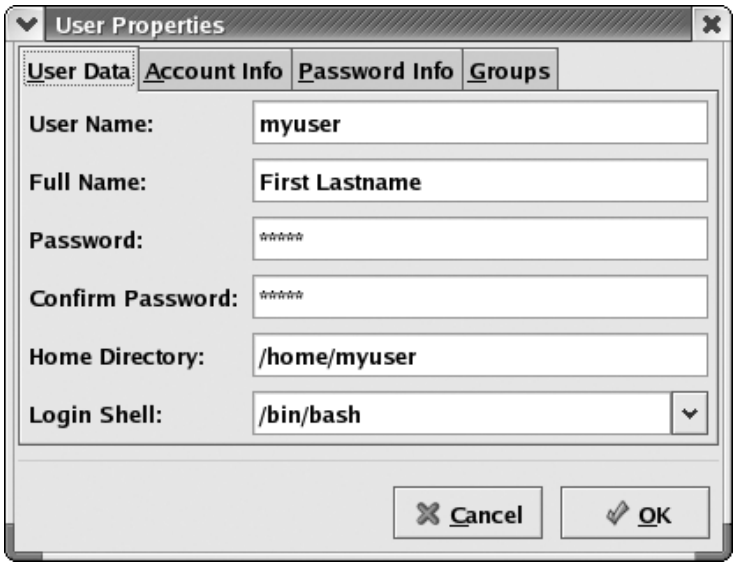

Figure 1-11

Figure 1-12

If the box next to Local password is locked is checked, the user will not be able to log in because the user's current password will not be correct. This is a good way to temporarily lock a user out of the system.

The Password tab, as shown in Figure 1-13, contains options related to the user's password. The date and time the user's password was last changed are displayed toward the top of the window, right underneath the tabs. If the user's password should expire after an amount of time, the box next to Enable password expiration should be checked.

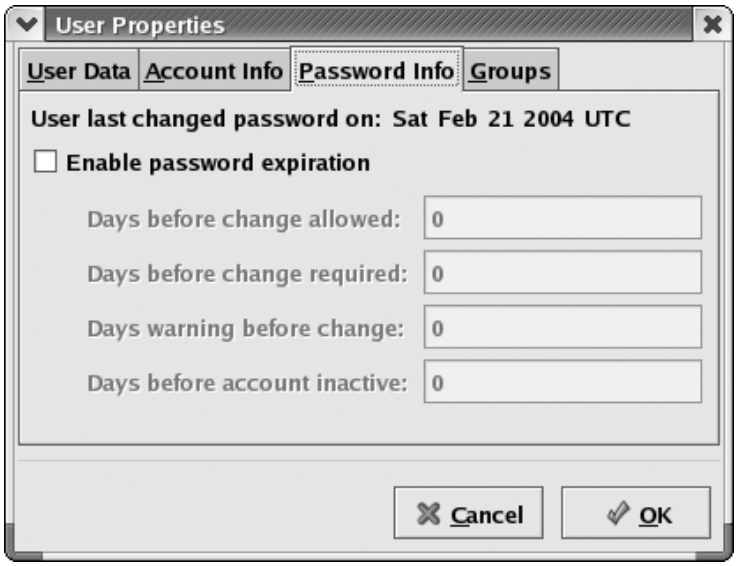

Figure 1-13

The Password tab also provides the ability for other fine-grained control over the user's password expiration. For instance the Days before change allowed field allows a period of time to be defined during which the user cannot change his or her new password. Many administrators take advantage of this feature because it prevents a user from changing an expired password to a new one, then changing it back to the old one the next day.

The Days before change required field effectively defines the maximum time that a password is valid. If the value is 60, users will be able to use their password for 60 days before the system requires them to change it to a new one.

If the Days warning before change value is not zero, users will be warned of the upcoming requirement for a new password. This allows a user to think of a new password, and might not be a bad idea because users will be more apt to come up with a good password and not forget it if they have a few days to think about it. If the value is zero, users will not be warned prior to the system prompting them for a new password.

A user's account can be deactivated if the user's password has not been changed. The Days before account inactive field defines the number of days that will pass before users' accounts are disabled if they haven't changed their expired password.

The Groups tab, as shown in Figure 1-14, provides a way for a user to be added to one or more groups.

The Primary Group defines the user's default group. This is the group that the user belongs to when the user logs into the server.

A user may be added to groups to allow the user to access specific devices, files, and directories.

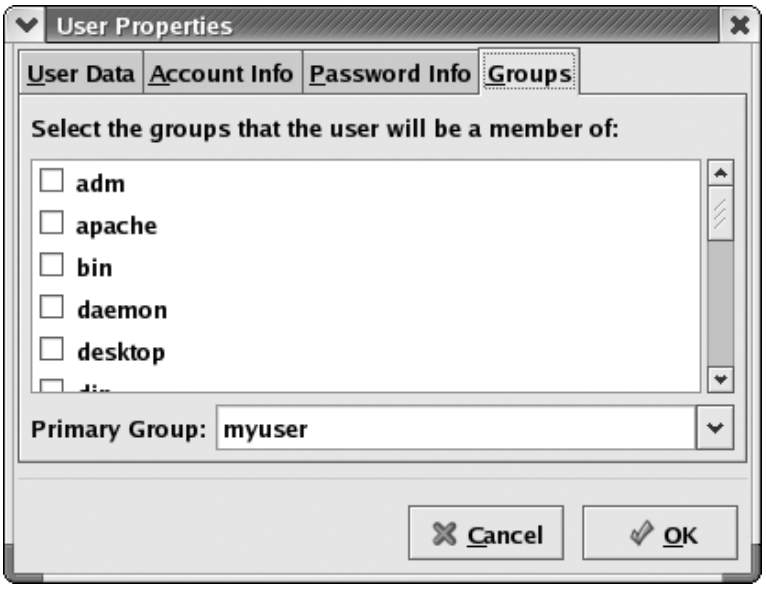

Figure 1-14

Modifying Users with usermod

User's properties such as the user's login shell, initial login group, expiration date, or even the username (referred to as login name in the `usermod` man pages) can also be changed on the command line using the `usermod` program. The `usermod` command does not allow password expiration policies to be set as on the Password tab of the GUI shown in the last section. Instead, the `passwd` command is used to set expiration policies for a user's password.

Some users prefer one shell over another, for instance `csh` over `sh`, and may request that their default login shell be changed. The available login shells are listed in the `/etc/shells` file, and any one of the shells listed in this file can be used for a user's default login shell. Many programs, such as `ftp` or `telnet` do not allow a user to log in if the user's shell is set to one that is not in the `/etc/shells` file. To change the login shell for the user `jdoe` to `/bin/csh`, type:

```
# usermod -s /bin/csh jdoe
```

The user's initial login group is used for security purposes and can be changed by using the `-g` parameter. For instance, if a user whose username is `jdoe` needs the initial login group needs to be changed to the `users` group, type:

```
# usermod -g users jdoe
```

A user's account can also be set to expire on a certain date. This is very useful for temporary user accounts, such as contractors or temps, because a date can be set when the account will expire and no longer be active. The date is specified in YYYY-MM-DD format. An administrator doesn't have to worry

about forgetting to disable the account later. The following command will set the user `jdoe`'s account to expire at the end of the year 2005:

```
# usermod -e 2005-12-31 jdoe
```

When an attempt is made to log into this account, the following message will be shown to the user:

```
Password:
Your account has expired; please contact your system administrator
```

To lock an account using `usermod`, pass the `-L` parameter to the `usermod` command. The `-U` parameter unlocks the account. The `-L` parameter simply modifies the encrypted password so that any login attempts result in a login failure, so the user will see the same behavior as he or she would have after typing in the wrong password. The system does not explicitly tell the user that the account has been locked. The `-U` parameter simply undoes the changes to the encrypted password made by `-L` so that the original password and login will work as they did originally.

More Information

More information about the `usermod` command can be found by typing `man usermod` or `info usermod` at the terminal, or by viewing `usermod` in the GNOME Help Browser.

Deleting a User with the GUI

To delete a user using the Users and Groups utility, highlight the user in the utility and click the Delete button or select Delete from the File menu. Before the user is deleted, a window will appear to confirm the action, as shown in Figure 1-15.

Figure 1-15

Deleting a User with userdel

The `userdel` command has only one option, `-r`, which, when executed, causes the given user's home directory and all files inside it will to be permanently removed. Use this command with extreme care, as with many of the other command-line programs; the results of this command are permanent and cannot be undone. To delete the user `jdoe` and all of the user's files inside the home directory, including the user's home directory itself, type:

```
# userdel -r jdoe
```

Chapter 1

Managing Files and Directories

Like many other tasks, Linux has something for everybody when it comes to managing files. Administrators that have Microsoft Windows experience will pick up the functionality of Nautilus fairly easily—files and directories can be dragged and dropped between locations, and files and folders when deleted are optionally moved to a Trash area, where they may be undeleted if necessary.

For administrators who have a Unix background, the command line will be a familiar way to manage files and directories. The familiar commands such as `ls`, `rm`, and `cp` support most of the same parameters as file management utilities in commercial Unixes such as AIX, HP-UX, and Solaris.

Listing Files and Directories

The `ls` command is used to list the contents of directories. When no options are used, the `ls` command will display a listing like this:

```
directory1 file1.txt file2.txt
```

Just the names of the files and directories are listed, going across the width of the terminal window. The `ls` command supports many options, both Portable Operating System Interface for UNIX (POSIX) options and GNU options. POSIX options are standard options that work across varying flavors of Unixes, so scripts are easily portable between various Unixes. More information about the parameters can be found by typing `man ls` or `info ls` on the command line.

To list the files and directories in a single column with more details (permissions, the owner and group, the size, and the date last modified), use the `-l` option. Typing `ls -l` on the same directory yields the following output:

```
drwxr-x---   2 jdoe   users              48 Feb   8 18:03 directory1
-rw-r-----   1 jdoe   users               0 Feb   8 17:53 file1.txt
-rw-r-----   1 jdoe   users               0 Feb   8 17:53 file2.txt
```

The first `d` on the first line shows that the listing for `directory1` is for a directory—regular files will have a "`-`" as the first character in the listing.

Directory names may be passed to `ls` on the command line after all of the options. For instance, to view the files in `directory1` above, type `ls directory1` at the command line.

More Information

For more information about the `ls` command, type `man ls` or `info ls` at the command line, or use the GNOME Help Browser.

Copying Files with Nautilus

Files may be copied with Nautilus by using a copy-and-paste method or simply by dragging and dropping a file onto a folder or between Nautilus windows.

To use the copy-and-paste method, open the context-sensitive menu by right-clicking on a file or folder in the main Nautilus window. Select Copy File from the menu. The file can also be copied by selecting a file and by using Ctrl+C on the keyboard. If the status bar is visible, the text in the status bar will change to indicate that the file has been copied. Now that the file has been copied, it can be pasted into the same

window, in a new window, or into a folder. To paste the file into the same folder or into a folder in a different window, select Paste Files from the context-sensitive menu or use the Ctrl+V keyboard combination. If the file or folder is pasted into the same folder, the name will be changed to show that the file is a copy. The files can also be pasted into a folder by highlighting a folder and selecting Paste Files into Folder from the context-sensitive menu.

To copy a file using the drag-and-drop method, select the file, and holding down the middle mouse button, drag the file from one folder to another. When you release the middle mouse button, select Copy here from the menu. Or, hold down the Ctrl key while dragging the file from one location to the next.

Copying Files and Directories with the Command Line

The `cp` command is used to copy files and directories, with the option of renaming them during the copying process. To read a full list of all of the parameters that `cp` supports, type `man cp` or `info cp` at the command line.

To copy a single file to another location using `cp`, pass the source file as the first parameter and the destination file as the second parameter. The following command:

```
# cp test.txt test2.txt
```

will copy the file named `test.txt` to the same directory, naming the new copy of the file `test2.txt`. When the last parameter given to `cp` is the name of a directory, the command will copy the first file into the directory without altering the file's name. The following command:

```
# cp test.txt mydir
```

will copy the `test.txt` file to `mydir/test.txt`, assuming that `mydir` is the name of a directory.

More than one file may be copied at a time by passing in more than one file name. However, when copying more than one file at a time, the last parameter must be the name of a directory. The following command will copy `test1.txt, test2.txt`, and `test3.txt` into the `mydir` directory:

```
# cp test1.txt test2.txt test3.txt mydir
```

The `cp` command has a parameter `-i` that tells `cp` to ask before copying over an existing destination file. It is a good idea to alias `cp` to `cp -i` in the default profile (see the "Building Default Profiles" section earlier in this chapter) in an attempt to prevent accidentally overwriting files. When `cp` is aliased to `cp -i`, the `-i` parameter does not have to be declared each time `cp` is run from the command line. To run `cp` without using the alias, use `\cp` at the command line.

The following is an example of the prompt that appears when a destination file already exists:

```
# cp file.txt file2.txt
cp: overwrite 'file2.txt'? y
```

Answering with anything except a `y` or `yes` here will cause `cp` to cancel the copying of the file.

Moving Files with Nautilus

Files can by moved in Nautilus using methods very similar to those used for copying files. Instead of using Copy to copy the file, use Cut from the context-sensitive menus. Once files are cut, they may be pasted into folders by selecting Paste Files from the context-sensitive menu.

The keyboard combination Ctrl-X may be used in place of Ctrl-V to move a file. The status bar, if visible, will change to indicate that the file will be moved once it is pasted into a different location.

Files may be moved by dragging and dropping the files from one folder to another.

Moving Files and Directories with the Command Line

The `mv` command is not only used to move files from one location to another, but also can be used to rename a command. The usage of the `mv` command is very similar to that of the `cp` command. Files can be specified on the command line, with the last argument being the name of a destination directory if more than one destination file is specified.

Like `cp`, the `mv` command supports the `-i` parameter that tells `mv` to prompt before replacing a destination file of the same name. In the root user's profile on my servers, one author also aliases `mv` to `mv -i` to guard against making easy mistakes due to typos. The `-f` parameter tells `mv` to force the move, even if the destination file already exists. To override the alias, use the "\" character in front of `mv`, like this:

```
# \mv file.txt file2.txt
```

With this syntax, the alias `mv -i` will be ignored, and the file will be replaced without prompting the user.

The following command would be used to move a file named `file.txt` to the `mydir` directory:

```
# mv file.txt mydir
```

The usage of `mv` to rename `file.txt` to `newfile.txt` is:

```
# mv file.txt newfile.txt
```

The `-v` parameter tells `mv` to be verbose, which means that the name of each file that is being moved is printed to the console.

Deleting Files with Nautilus

Files and folders may be deleted in Nautilus by using the shortcut menus, by using the keyboard, or by dragging and dropping the files.

By default in Nautilus, the Delete item is not visible in the shortcut menus or in the Edit menu. Instead, a menu item labeled Move to Trash enables files to be moved to a temporary space before being deleted permanently. Windows administrators will recognize this functionality as being very similar to that of the Windows Recycle Bin. To move a file to the Trash folder, highlight the file and select Move to Trash from the context-sensitive menu or from the Edit menu. If the Trash icon is visible on the desktop, the file or folder can also be dragged and dropped onto the Trash icon to move the file to the Trash. Files and folders in the Trash are removed permanently by selecting Empty Trash from the Trash's shortcut menu.

Nautilus can be configured to display a Delete command in the context-sensitive menus and in the Edit menu. This Delete command, when visible, removes a file permanently from the file system without first moving it to the Trash. The Delete command can be turned on by selecting Preferences from the Edit menu and by checking Include a Delete command that bypasses Trash under the Behavior tab.

Using the keyboard, a file may be moved to the Trash folder by using the Delete key or by using the Ctrl+T combination. A file may be permanently deleted without moving it to the trash by using the Shift+Delete combination while a file is highlighted. The Shift+Delete combination only works if the option to include the delete command is checked in the preferences (see previous paragraph).

Deleting Files and Directories with the Command Line

The `rm` command is used to delete files and directories, and is similar to `cp` and `mv` in the way it accepts file names and directories. The `rm` command removes files permanently from disk—there is no temporary folder in which files are moved before they are permanently deleted. Since this is the case, it is important to be very careful when using the `rm` command at the command line. Type slowly, and *very seriously* consider using the `rm -i` alias in the user's profile (see the "Building Default Profiles" section for more information) to add an interactive step.

To illustrate the importance of using interactive mode with `rm`, consider what would happen if `rm * tmp` were typed in instead of `rm *.tmp`. Note that there is a space between the wildcard character and `tmp` in the first example, which means that `rm` will interpret each as an argument by itself. The shell will expand the wildcard to *every file in the current directory,* then will try to remove a file named `tmp`.

Renaming a File with Nautilus

Files and folders can be renamed in Nautilus by selecting a file and using either the menus, the Properties dialog box, or keyboard shortcuts.

To rename a file using the menus, select a file and either use the context-sensitive menu or the Edit menu to select the Rename menu item. The name of the file will be highlighted, and a box will appear around the name of the file. Notice that if the file has an extension, the extension will not be highlighted when renaming a file. However, the entire file name can by selected by triple-clicking on it. Double-clicking will select the file's name up to the extension, or the extension only depending on which item is double-clicked.

A file may also be renamed by using the Properties of the file. Open the Properties window by selecting Properties from the context-sensitive menu or from the File menu when a file or folder is highlighted. The Properties window will open with the name of the file already highlighted. The file's name can by changed by typing a new name into the Name field.

The keyboard shortcut for renaming a file is the F2 key. When a file is highlighted, pressing the F2 key is just like using the Rename menu item. To cancel the renaming of the file, use the Sec key. To commit the file name change, use the Enter key.

Creating Symbolic Links with Nautilus

A feature of file management that may be new to most Windows administrators but familiar to Unix administrators is symbolic links for files and folders. A symbolic link is a pseudofile in the sense that it represents the file to which it is linked. A symbolic link is a pointer to a file that allows a symbolic link to

be opened or executed as if it were the file itself. Symbolic links of files may be created in Nautilus using methods similar to those used for other types of file management.

To create a symbolic link using the menus, select a file or folder and click the "Make Link" menu item from the context-sensitive menu or from the Edit menu. The keyboard shortcut for creating a symbolic link is the Ctrl+L combination, which can be used when a file is selected.

A symbolic link to a file may also be created by dragging and dropping a file while holding down the middle mouse button down, then choosing Link here from the menu that appears when the mouse button is released.

Creating Symbolic Links with the Command Line

The `ln` command is used to create a symbolic link at the command line. The `--s` parameter tells `ln` to create the symbolic link. To link file `/home/user/test` to `/home/user/testlnk`, type:

```
ln --s /home/user/test /home/user/testlnk
```

The new file, `/home/user/testlnk`, will be a pointer to the file that already exists. The link may be deleted without deleting the original file.

Opening a File with Nautilus

A file can be opened in Nautilus with the application that is configured for the file's type in the File Types and Programs preference tool. Files may also be opened with a particular application from one of the menus, or by dragging and dropping a file onto an application's icon.

The File Types and Programs preference utility can be opened by typing `preferences:///Advanced` into the Nautilus location bar and pressing Enter, then by double-clicking on the icon labeled, File types and programs. The other method of modifying a file association is by clicking the Open With menu item in either the context-sensitive menu or the File menu, then selecting the Other Application menu item. Click the Go There button under the File Types and Programs section in the Open with Other Application window.

Most files that you will be opening already have programs associated with their file type, so opening a file is as simple as double-clicking the file's icon in Nautilus. Some files will open in Nautilus's main panel for viewing, including plain text files, PDF files, and some image files.

Examining Processes

The processes that are running on a system can be audited by using both GUI tools and command-line tools. The GUI process viewer is similar to the Task Manager in the Microsoft Windows operating system, where processes can be ended if they need to be. At the command line, `ps` can be used to list the running processes and when used with other programs such as `grep`, processes can easily be located.

Using the GUI

Processes can be monitored in the desktop environment by using the System Monitor in the System Tools. Under the Process Listing tab, a listing similar to the one seen by using the top command can be found.

The processes are listed by default in the order in which they are using memory. The process list can be sorted by clicking on the column headers, so to sort the list to see which processes are using the most CPU capacity, click the % CPU column heading. To end a single process or process tree, click the End Process button.

Using ps

The `ps` command lists processes and can display a lot of information about them, such as the process ID (PID), the amount of CPU that the process is using, the amount of memory that the process is using, and the name of the command that is running. The `ps` command prints the process information out to `STDIN`, so the output of `ps` can be processed by other commands on the command line such as grep.

In combination with `grep`, `ps` can be used to isolate a process to get more information about it. Here is an example of using `ps` and `grep` together to find the PIDs (process IDs) of all of the instances of Nautilus:

```
#ps -e | grep -i Nautilus
```

The `-e` parameter used with `ps` tells `ps` to select all processes for all users. If the `-e` parameter is not specified, the `ps` command will only show the current shell's child processes. Using the `-i` parameter with `grep` tells `grep` to ignore case, which is important because the name of the process is actually "nautilus," with no capitalization. When the command above is run, the output will look like this:

```
4949 ?           00:00:02 nautilus
4956 ?           00:00:00 nautilus
4957 ?           00:00:00 nautilus
4958 ?           00:00:00 nautilus
4959 ?           00:00:00 nautilus
```

The first column of the output contains the process identifier, or PID, of the process. The third column shows the amount of processor time that the process has taken since it has been started, and the last column shows the process name.

Even more information about processes can be shown by using the `-f` parameter with `ps`, including the process' parent process identifier, the user under which the process is running, the start time of the process, and the full command name, including parameters that were passed to the command.

The `-U` parameter displays the processes for a given user, which can be defined by the user ID or the user's login name. The following command:

```
# ps -U jdoe -f
```

will output the full information about processes being executed by the user `jdoe`. The output might look something like this:

```
UID        PID  PPID  C STIME TTY          TIME CMD
jdoe      4887  2229  0 09:01 ?        00:00:00 /usr/bin/gnome-session
jdoe      4898  4887  0 09:01 ?        00:00:00 /usr/bin/ssh-agent --
/etc/X11/gdm/gnomerc
jdoe      4900     1  0 09:01 ?        00:00:01 /usr/libexec/gconfd-2 5
```

The last process, `gconfd`, has a parent PID of 1. The `init` process, which is the process that starts all other processes, has the PID of 1, so this means that the parent of the last process listed above is `init`. It is not a good idea to try to kill or restart the `init` process as the system will shut down or restart.

Using top

The `top` command shows a lot of information about processes and the resources that are used by them. It also has a summary that displays the amount of memory, swap space, and CPU utilization available on the server. Unlike `ps`, which is a static list of information about processes, `top` continually refreshes until it is ended. Since it shows process and system information dynamically, it is an ideal command to use for monitoring a server or watching a single process.

By default, the output of `top` sorts processes from top to bottom with the processes that are using the most resources at the top of the list.

Summary

RHEL 3 is a multiuser, enterprise-level Linux distribution that comes with many different tools that can be used to accomplish tasks. Specifically, RHEL 3 offers much to administrators in the form of GUI tools for user, file, and process management.

The command line enables the administration of a system remotely without the need of a GUI, and virtual terminals allow more than one command-line session to be open at the same time on the console. Also, GUI applications can be displayed on a different system over the network, often referred to as *exporting a display*. This allows GUI applications to be run remotely.

Users can be added, removed, and modified by using both the GUI and the command line. Default templates can be made for profiles, which makes creating new users easier. The command-line tools allow users to be administered remotely.

Using the command line, files can be easily listed, copied, moved, deleted, edited, and viewed remotely or locally without a GUI. However, GUI applications are also provided if the desktop environment is installed, to accomplish these same tasks without having to type any commands.