# Unit 10: Linux System Administration

CONTENTS



## Objectives

After studying this unit, you will be able to

- Know about the system administrator and superuser
- Understand the rescue mode
- Understand the SELinux
- Understand the system operations and system administration utilities
- Understand how to set up a server
- know important files and directories in Linux
- Understand the file types and file systems
- Know how to back up files and schedule tasks
- Understand how to configure user and group accounts, system reports and parted

# Introduction

A well-maintained system:

- Runs quickly enough so users do not get too frustrated waiting for the system to respond or complete a task.
- Has enough storage to accommodate users' reasonable needs.
- Provides a working environment appropriate to each user's abilities and requirements.
- Is secure from malicious and accidental acts altering its performance or compromising the security of the data it holds and exchanges with other systems.
- Is backed up regularly, with recently backed-up files readily available to users.
- Has recent copies of the software that users need to get their jobs done.
- Is easier to administer than a poorly maintained system.

## 10.1 System Administrator and Superuser

A system administrator should be available to help users with all types of system-related problems from logging in to obtaining and installing software updates to tracking down and fixing obscure network issues. Much of what a system administrator does is work that ordinary user do not have permission to do. When performing one of these tasks, the system administrator logs in as root to have system wide powers that are beyond those of ordinary users: A user with root privileges is referred to as **Super user**. The username is root by default.

Superuser has the following powers:

- Some commands, such as those that add new users, partition hard drives, and change system configuration, can be executed only by root.
- Superuser can use certain tools, such as sudo, to give specific users permission to perform tasks that are normally reserved for Superuser.
- Read, write, and execute file access and directory access permissions do not affect root: Superuser can read from, write to, and execute all files, as well as examine and work in all directories.
- Some restrictions and safeguards that are built into some commands do not apply to root. For example, root can change any user's password without knowing the old password.

When you are running with root (Superuser) privileges, the shell by convention displays a special prompt to remind you of your status. By default, this prompt is or ends with a pound sign (#).

### Gain Superuser Privileges

You can gain or grant Superuser privileges in several ways:

1) When you bring the system up in single-user mode, you are Superuser.

2) Once the system is up and running in multiuser mode, you can log in as root. When you supply the proper password, you will be Superuser.

3) You can give an su (substitute user) command while you are logged in as yourself and, with the proper password, you will have Superuser privileges.

4) You can use sudo selectively to give users Superuser privileges for a limited amount of time on a per-user and per-command basis. The sudo utility is controlled by the /etc/sudoers file, which must be set up by root.

5) Any user can create a setuid (set user ID) file. Setuid programs run on behalf of the owner of the file and have all the access privileges that the owner has. While you are running as Superuser, you can change the permissions of a file owned by root to setuid. When an ordinary user executes a file that is owned by root and has setuid permissions, the program has full root privileges.

6) Some programs ask you for a password (either your password or the root password, depending on the command and the configuration of the system) when they start. When you provide the root password, the program runs with root privileges.

## 10.2 System Administration Tools

Many tools can help you be an efficient and thorough system administrator.

- su: Gives You Another User's Privileges
- console helper: Runs Programs as root
- kill: Sends a Signal to a Process

### su: Gives You Another User's Privileges

The su (substitute user) utility can create a shell or execute a program with the identity and permissions of a specified user. Follow su on the command line with the name of a user; if you are working with root privileges or if you know the user's password, you take on the identity of that user. When you give an su command without an argument, su defaults to Superuser so that you take on the identity of root (you have to know the root password). It is better to use /bin/su which is the official version of su to avoid the trouble. When you give an su command to become Superuser, you spawn a new shell, which displays the # prompt. You return to your normal status (and your former shell and prompt) by terminating this shell: Press CONTROL-D or give an exit command.

### Console helper: Runs Programs as root

The console helper utility can make it easier for someone who is logged in on the system console but not logged in as root to run system programs that normally can be run only by root.

### kill: Sends a Signal to a Process

The kill built in sends a signal to a process. This signal may or may not terminate (kill) the process, depending on which signal is sent and how the process is designed.

## 10.3 Rescue Mode

Rescue mode is an environment you can use to fix a system that does not boot normally. To bring a system up in rescue mode, boot the system from the first installation CD, the Net Boot CD, or the install DVD. From the install DVD, select Rescue installed system from the Welcome menu. From the first installation CD and the Net Boot CD, enter the rescue (FEDORA) or boot rescue (RHEL) boot parameter. In rescue mode, you can change or replace configuration files, check, and repair partitions using fsck, rewrite boot information, and more. The rescue screen first asks if you want to set up the network interface. This interface is required if you want to copy files from other systems on the LAN or download files from the Internet.

### Avoiding a Trojan Horse

A Trojan horse is a program that does something destructive or disruptive to a system while appearing to be benign. As an example, you could store the following script in an executable file named mkfs:

```
while true
do
echo 'Good Morning Mr. Jones. How are you? Ha Ha Ha.' > /dev/console
done
```

If you are running as Superuser when you run this command, it would continuously write a message to the console. If the programmer were malicious, it could do worse. The only thing missing in this plot is access permissions. A malicious user could implement this Trojan horse by changing Superuser's PATH variable to include a publicly writable directory at the start of the PATH string. A good way to help prevent the execution of a Trojan horse is to make sure that your PATH variable does not contain a single colon (:) at the beginning or end of the PATH string or a period (.) or double colon (::) anywhere in the PATH string. This precaution ensures that you will not execute a file in the working directory by accident.

## 10.4 Security Enhanced Linux

Traditional Linux security, i.e., DAC is based on users and groups. Because a process run by a user has access to anything the user has access to, fine-grained access control is difficult to achieve. SELinux was developed by the U.S. NSA, implements MAC in the Linux kernel. MAC enforces security policies that limit what a user or program can do. It defines a security policy that controls some or all objects, such as files, devices, sockets, and ports, and some or all subjects, such as processes. Using SELinux, you can grant a process only those permissions it needs to be functional, following the principle of least privilege. MAC is an important tool for limiting security threats that come from user errors, software flaws, and malicious users. The kernel checks MAC rules after it checks DAC rules.

### States/Modes of SELinux

SELinux can be in one of three states (modes):

- **Enforcing—** This is the default state.
- **Permissive—** This is the diagnostic state.
- **Disabled—** No policy.

### Policies of SELinux

SELinux implements one of the following policies:

- **Targeted—**Applies SELinux MAC controls only to certain (targeted) processes (default).
- **MLS—**Multilevel Security protection.
- **Strict—**Applies SELinux MAC controls to all processes (RHEL).

### Turning off SELinux

There are two ways to disable SELinux: You can modify the /etc/selinux/config file so that it includes the line SELINUX=disabled and reboot the system, or you can use system-config-selinux.

### config: The SELinux Configuration File

The /etc/selinux/config file, which has a link at /etc/sysconfig/selinux, controls the state of SELinux on the local system. Although you can modify this file, it may be more straightforward to work with system-config-selinux. In the following example, the policy is set to targeted, but that setting is of no consequence because SELinux is disabled:

$ **cat /etc/selinux/config**

# This file controls the state of SELinux on the system.

# SELINUX= can take one of these three values:

# enforcing - SELinux security policy is enforced.

# permissive - SELinux prints warnings instead of enforcing.

# disabled - SELinux is fully disabled.

SELINUX=disabled

# SELINUXTYPE= type of policy in use. Possible values are:

# targeted - Only targeted network daemons are protected.

# strict - Full SELinux protection.

SELINUXTYPE=targeted

To put SELinux in enforcing mode, change the line containing the SELINUX assignment to SELINUX=enforcing. Similarly, you can change the policy by setting SELINUXTYPE.

## getenforce, setenforce, and sestatus: Work with SELinux

The getenforce and setenforce utilities report on and temporarily set the SELinux mode. The sestatus utility displays a summary of the state of SELinux:

**# getenforce**

Enforcing

**# setenforce permissive**

**# sestatus**

SELinux status: enabled

SELinuxfs mount: /selinux

Current mode: permissive

Mode from config file: enforcing

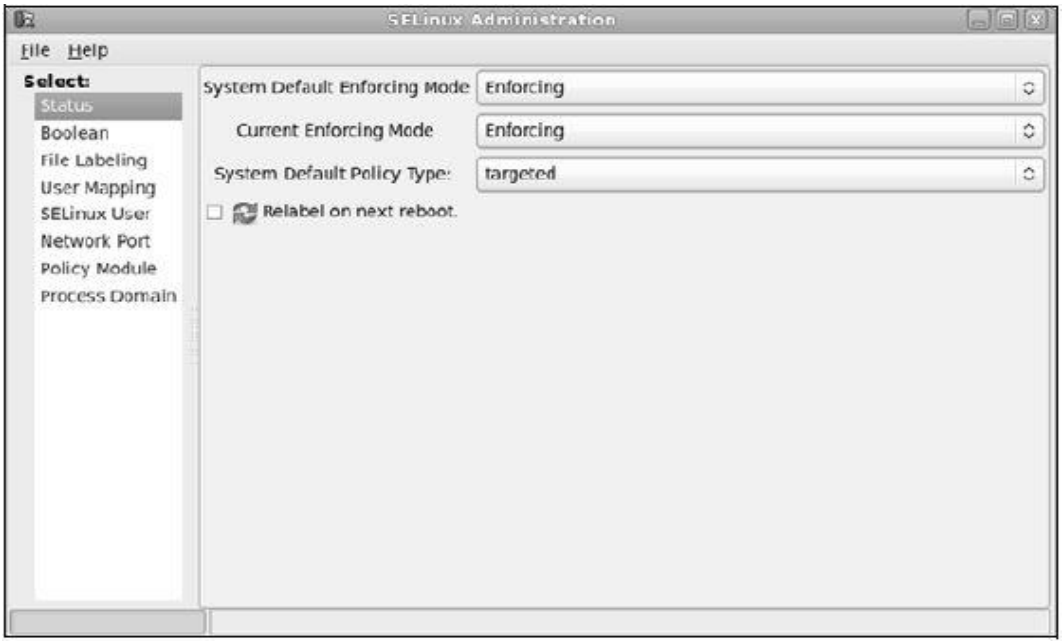
Policy version: 24

Policy from config file: targeted

### Setting the Targeted Policy with system-config-selinux

The system-config-selinux utility displays the SELinux Administration window, which controls SELinux. To run this utility, enter system-config-selinux from a command line in a graphical environment or select Main menu: System□Administration□SELinux Management.

### SELinux Administration window




With Status highlighted on the left side of the SELinux Administration window, choose Enforcing (default), Permissive, or Disabled from the drop-down list labeled System Default Enforcing Mode. The mode you choose becomes effective next time you reboot the system. You can use the drop-down list labeled Current Enforcing Mode to change between Enforcing and Permissive modes immediately. When you change the mode using this list, the system resumes the default mode when you reboot it. To modify the SELinux policy, highlight Boolean on the left side of the SELinux Administration window and scroll through the list of modules. To find modules that pertain to NFS, type nfs in the text box labeled Filter and then press RETURN. The SELinux Administration window displays all modules with the string nfs in their descriptions. The modules with tick marks in the Active column are in use.

## 10.5 Run levels

| Number | Name | Login | Level | Filesystems |
|---|---|---|---|---|
| 0 | Halt | | | |
| 1 | Single user | Textual | Down | Mounted |
| 2 | Multiuser without NFS | Textual | Up | Mounted |
| 3 | Multiuser | Textual | Up | Mounted |
| 4 | User Defined | | | |
| 5 | Multiuser with X | Graphical | Up | Mounted |
| 6 | Reboot | | | |

- **Default Run level**: By default, Fedora systems boot to graphical multiuser mode (runlevel 5).
- **runlevel utiity**: The runlevel utility displays the previous and current runlevels. This utility is a transitional tool; it provides compatibility with SysVinit. In the following example, the N indicates that the system does not know what the previous runlevel was and the 5 indicates that the system is in multiuser mode.

  $ runlevel

  N 5
- **telinit utility**: The telinit utility allows a user with root privileges to bring the system down, reboot the system, or change between recovery (single-user) and multiuser modes. The telinit utility is a transitional tool; it provides compatibility with SysVinit. The format of a telinit command is: telinit runlevel.
- **Recovery mode and the root password:** When the system enters recovery (single-user) mode, init requests the root password before displaying the root prompt. When the system enters graphical multiuser mode, it displays a graphical login screen.

## 10.6 Booting the System

Booting a system is the process of reading the Linux kernel into system memory and starting it running. As the last step of the boot procedure, Linux runs the init program as PID number 1. The init program is the first genuine process to run after booting and is the parent of all system processes. (That is why when you run as root and kill process 1, the system dies.)

- **initdefault**: The initdefault entry in the /etc/inittab file tells init which runlevel to bring the system to. Set initdefault to 3 to cause the system to present a text login message when it boots; set it to 5 to present a graphical login screen (default).
- **init daemon:** As the last step of the boot procedure, Linux starts the init daemon as PID number 1. The init daemon is the first genuine process to run after booting and is the parent of all system processes.

### Init Scripts: Start and Stop System Services

The first script that runs is /etc/rc.d/rc.sysinit, which performs basic system configuration, including setting the system clock, hostname, and keyboard mapping; setting up swap partitions; checking the filesystems for errors; and turning on quota management.

#### service: Configures Services I

Fedora/RHEL provides service, a handy utility that can report on or change the status of any of the system services in **/etc/rc.d/init.d.**

#### system-config-services: Configures Services II

The system-config-services utility displays the Service Configuration window. This utility has two functions: It turns system services on and off immediately, and it controls which services are stopped and started when the system enters and leaves runlevels 2–5. The system-config-services utility works with many of the services listed in /etc/rc.d/init.d as well as with those controlled by xinetd and listed in /etc/xinetd.d (or specified in /etc/xinetd.conf). To run system-config-services, enter system-config-services from a command line in a graphical environment or select Main menu: System | Administration | Services.

#### chkconfig: Configures Services III

The chkconfig character-based utility duplicates much of what the system-config services utility does: It makes it easier for a system administrator to maintain the /etc/rc.d directory hierarchy.

This utility can add, remove, list startup information, and check the state of system services. It changes the configuration only—it does not change the current state of any service.

### Single-User Mode

When the system is in single-user mode, only the system console is enabled. You can run programs from the console in single-user mode just as you would from any terminal in multiuser mode. The only difference is that few of the system daemons will be running. The scripts in /etc/rc.d/rc1.d are run as part of single-user initialization. With the system in single-user mode, you can perform system maintenance that requires file systems to be unmounted or that requires just a quiet system—no one except you using it, so that no user programs interfere with disk maintenance and backup programs.

### Going to Multiuser Mode

After you have determined that all is well with the filesystems, you can bring the operating system up to multiuser mode. When you exit from the single-user shell, init brings the system to the default run level—usually 5. Alternatively, you can give the following command in response to the Superuser prompt to bring the system to textual multiuser mode (use 5 to go to graphical multiuser mode): # /sbin/telinit 3. When it goes from single-user to textual multiuser mode, the system executes the K (kill or stop) scripts and then the S (start) scripts in /etc/rc.d/rc3.d.

### Graphical Multiuser Mode

Graphical multiuser mode is the default state for a Fedora/RHEL system. In this mode all appropriate filesystems are mounted, and users can log in from all connected terminals, dial-up lines, and network connections. All support services and daemons are enabled and running. Once the system is in graphical multiuser mode, a login screen appears on the console. Most systems are set up to boot directly to graphical multiuser mode without stopping at single-user mode.

## Logging In

- Textual login
- Graphical login

### Textual login

With a textual login, the system uses init, mingetty, and login to allow a user to log in; login uses PAM modules to authenticate users. The system is in multiuser mode, the Upstart init daemon is responsible for spawning a mingetty process on each of the lines that a user can use to log in.

### Graphical login

With a graphical login, the Upstart init daemon starts gdm (the GNOME display manager) by default on the first free virtual terminal, providing features similar to those offered by mingetty and login. The gdm utility starts an X server and presents a login window. The gdm display manager then uses PAM to authenticate the user and runs the scripts in the /etc/gdm/PreSession directory.

### Logging Out

When the system displays a shell prompt, you can either execute a program or exit from the shell. If you exit from the shell, the process running the shell dies and the parent process wakes up. When the shell is a child of another shell, the parent shell wakes up and displays a prompt. Exiting from a login shell causes the operating system to send Upstart a signal that one of its children has died. Upon receiving this signal, Upstart takes action based on the contents of the appropriate tty job definition file. In the case of a process controlling a line for a terminal, Upstart informs mingetty that the line is free for another user. When you are at runlevel 5 and exit from a GUI, the GNOME display manager, gdm, initiates a new login display.

### Bringing the System Down

The shutdown and halt utilities perform the tasks needed to bring the system down safely. These utilities can restart the system, prepare the system to be turned off, put the system in single-user mode, and, on some hardware, power down the system. The poweroff and reboot utilities are linked to halt. If you call halt when the system is not shutting down (runlevel 0) or rebooting (runlevel 6), halt calls shutdown. CONTROL-ALT-DEL: Reboots the System

### Crash

A crash occurs when the system stops suddenly or fails unexpectedly. A crash may result from software or hardware problems or from a loss of power. As a running system loses power, it may behave in erratic or unpredictable ways. In a fraction of a second, some components are supplied with enough voltage; others are not. Buffers are not flushed, corrupt data may be written to the hard disk, and so on. IDE drives do not behave as predictably as SCSI drives under these circumstances. After a crash, you must bring the operating system up carefully to minimize possible damage to the filesystems. On many occasions, little or no damage will have occurred.

## 10.7 System Administration Utilities

These utilities can help you perform system administration tasks.

### Fedora/RHEL configuration tools

Most of the Fedora/RHEL configuration tools are named system-config-*. These tools bring up a graphical display when called from a GUI; some display a textual interface when called from a non-GUI command line. Some, such as system-configfirewall-tui, use a name with a –tui extension for the textual interface.

| Tool | Function |
|---|---|
| **system-config-authentication** | Displays the Authentication Configuration window with three tabs.<br>**User Information tab**: It allows you to enable NIS, LDAP, Hesiod, and Winbind support.<br>**Authentication tab**: It allows you to work with Kerberos, LDAP, Smart Card, Fingerprint Reader, and Windbind.<br>**Options tab**: It allows you to use shadow and sha512 passwords as well as to enable other system options. |
| **system-config-bind** | Displays the Domain Name Service window. |
| **system-config-boot** | Allows you to specify a default kernel and timeout for the **grub.conf file** |
| **system-config-display** | Brings up the Display Settings window with three tabs: Settings, Hardware, and Dual Head. |

| | |
|---|---|
| **system-config-date** | Displays the Date/Time Properties window with two tabs: Date & Time and Time Zone.<br>**Date & Time:** You can set the date and time or enable NTP (Network Time Protocol) from the first tab.<br>**Time:** The Time Zone tab allows you to specify the time zone of the system clock or set the system clock to *UTC* |
| **system-config-firewall[-tui] (FEDORA)** | Displays the Firewall Configuration window |
| **system-config-httpd** | Displays the HTTP window with four tabs: Main, Virtual Hosts, Server, and Performance Tuning |
| **system-config-keyboard** | Displays the Keyboard window, which allows you to select the type of keyboard attached to the system. You use this utility to select the keyboard when you install the system. |
| **system-config-language** | Displays the Language Selection window, which allows you to specify the default system language from among those that are installed. You use this utility to select the system language when you install the system. |
| **system-config-lvm** | Displays the Logical Volume Management window, which allows you to modify existing logical volumes |
| **system-config-network[-tui]** | Displays the Network Configuration window |
| **system-config-network-cmd** | Displays the parameters that system-config-network uses. |
| **system-config-nfs** | Displays the NFS Server Configuration window |
| **system-config-packages (RHEL)** | Runs pirut |
| **system-config-printer** | Displays the Printer Configuration window, which allows you to set up printers and edit printer configurations |
| **system-config-rootpassword** | Displays the Root Password window, which allows you to change the root password. While logged in as root, you can also use passwd from a command line to change the root password. |
| **system-config-samba** | Displays the Samba Server Configuration window, which can help you configure Samba |

| | |
|---|---|
| **system-config-selinux (FEDORA)** | Displays the SELinux Administration window, which controls SELinux |
| **system-config-services** | Displays the Service Configuration window, which allows you to specify which daemons (services) run at each runlevel. |
| **system-config-soundcard (RHEL)** | Displays the Audio Devices window, which tells you which audio device the system detected and gives you the option of playing a sound to test the device |
| **system-config-users** | Displays the User Manager window, which allows you to work with users and groups |

## Command-Line Utilities

- **chsh:** Changes the login shell for a user. When you call chsh without an argument, you change your own login shell. Superuser can change the shell for any user by calling chsh with that user's username as an argument.
- **clear:** Clears the screen. You can also use CONTROL-L from the bash shell to clear the screen.
- **dmesg:** Displays the kernel ring buffer.
- **e2label:** Displays or creates a volume label on an ext2, ext3, or ext4 filesystem. An e2label command has the following format: e2label device [newlabel]

where device is the name of the device (e.g., /dev/hda2, /dev/sdb1, /dev/fd0) you want to work with. When you include the optional newlabel parameter, e2label changes the label on device to newlabel. Without this parameter, e2label displays the label. You can also create a volume label with the –L option of tune2fs.

- **mkfs:** Creates a new filesystem on a device. This utility is a front end for many utilities, each of which builds a different type of filesystem. By default, mkfs builds an ext2 filesystem and works on either a hard disk partition or a floppy diskette. Although it can take many options and arguments, you can use mkfs simply as # mkfs device

where device is the name of the device (e.g., /dev/hda2, /dev/sdb1, /dev/fd0) you want to make a file system on.

- **ping:** Sends packets to a remote system. This utility determines whether you can reach a remote system through the network and tells you how much time it takes to exchange messages with the remote system.
- **reset (link to tset):** Resets terminal characteristics. The value of the environment variable **TERM** determines how to reset the screen. The screen is cleared, the kill and interrupt characters are set to their default values, and character echo is turned on. When given from a graphical terminal emulator, this command also changes the size of the window to its default. The reset utility is useful to restore your screen to a sane state after it has been corrupted.
- **setserial:** Gets and sets serial port information. Superuser can use this utility to configure a serial port. The following command sets the input address of **/dev/ttys0 to 0x100,** the interrupt (IRQ) to 5, and the baud rate to 115,000 baud:

  **# setserial /dev/ttys0 port 0x100 irq 5 spd_vhi**

- **stat:** Displays information about a file or filesystem. Giving the –f (filesystem) option followed by the device name or mount point of a filesystem displays information about the filesystem including the maximum length of filenames.

$ **stat -f /dev/sda**

File: "/dev/sda"

ID: 0 Namelen: 255 Type: tmpfs

Block size: 4096 Fundamental block size: 4096

Blocks: Total: 121237 Free: 121206 Available: 121206

Inodes: Total: 121237 Free: 120932

- **umask:** shell builtin that specifies a mask the system uses to set up access permissions when you create a file. A umask command has the following format: umask **[mask].**

-where mask is a three-digit octal number or a symbolic value such as you would use with chmod. The mask specifies the permissions that are not allowed.

When mask is an octal number, the digits correspond to the permissions for the owner of the file, members of the group the file is associated with, and everyone else. Because mask specifies the permissions that are not allowed, the system subtracts each of the three digits from 7 when you create a file. A mask that you specify using symbolic values indicates the permissions that are allowed.

- **uname:** Displays information about the system. Without any arguments, this utility displays the name of the operating system (Linux). With a –a (all) option, it displays the operating system name, hostname, version number and release date of the operating system, and type of hardware you are using:

  $ uname -a

  Linux F12 2.6.31.6-145.fc12.i686.PAE #1 SMP Sat Nov 21 16:12:37 EST 2009 i686 athlon i386 GNU/Linux

## 10.8 Standard Rules in Configuration Files

Most configuration files, which are typically named *.conf, rely on the following conventions:

1) Blank lines are ignored.
2) A # anywhere on a line starts a comment that continues to the end of the line. Comments are ignored.
3) When a name contains a SPACE, you must quote the SPACE by preceding it with a backslash (\) or by enclosing the entire name within single or double quotation marks.
4) To make long lines easier to read and edit, you can break them into several shorter lines. Break a line by inserting a backslash (\) immediately followed by a NEWLINE (press RETURN in a text editor).

### Specifying Clients

Some common ways to specify a host or a subnet:

| Client name pattern | Matches |
|---|---|
| n.n.n.n | One IP address. |
| name | One hostname, either local or remote. |
| Name that starts with . | Matches a hostname that ends with the specified string. For example, .tcorp.com matches the systems kudos.tcorp.com and speedy.tcorp.com, among others. |

| | |
|---|---|
| IP address that ends with . | Matches a host address that starts with the specified numbers. For example, 192.168.0. matches 192.168.0.0 – 192.168.0.255. If you omit the trailing period, this format does not work. |
| Starts with @ | Specifies a netgroup. |
| n.n.n.n/m.m.m.m or n.n.n.n/mm | An IP address and subnet mask specify a subnet. |
| Starts with / | An absolute pathname of a file containing one or more names or addresses as specified in this table. |

| Wildcard | Matches |
|---|---|
| * and ? | Matches one (?) or more (*) characters in a simple hostname or IP address. These wildcards do not match periods in a domain name. |
| ALL | Always matches. |
| LOCAL | Matches any hostname that does not contain a period. |

| Operator | Function |
|---|---|
| EXCEPT | Matches anything in the preceding list that is not in the following list. For example, a b c d EXCEPT c matches a, b, and d. Thus you could use 192.168. EXCEPT 192.168.0.1 to match all IP addresses that start with 192.168. except 192.168.0.1. |

## Specifying a Subnet

When you set up a server, you frequently need to specify which clients are allowed to connect to the server. Sometimes it is convenient to specify a range of IP addresses, called a subnet. Usually, you can specify a subnet as

n.n.n.n/m.m.m.m

or

n.n.n.n/maskbits

where n.n.n.n is the base IP address and the subnet is represented by m.m.m.m (the subnet mask) or maskbits (the number of bits used for the subnet mask). Example: 192.168.0.1/255.255.255.0 represents the same subnet as 192.168.0.1/24. In binary, decimal 255.255.255.0 is represented by 24 ones followed by 8 zeros. The /24 is shorthand for a subnet mask with 24 ones.

### Different ways to represent a subnet

| Bits | Mask | Range |
|---|---|---|
| 10.0.0.0/8 | 10.0.0.0/255.0.0.0 | 10.0.0.0 – 10.255.255.255 |
| 172.16.0.0/12 | 172.16.0.0/255.240.0.0 | 172.16.0.0 – 172.31.255.255 |
| 192.168.0.0/16 | 192.168.0.0/255.255.0.0 | 192.168.0.0 – 192.168.255.255 |

### rpcinfo: Displays Information About rpcbind

Fedora uses the rpcbind daemon while RHEL uses portmap for the same purpose. The rpcinfo utility displays information about programs registered with rpcbind and makes RPC calls to programs to see if they are alive. The rpcinfo utility takes the following options and arguments:

rpcinfo –p **[host]**

rpcinfo [–n **port] –u | –t host program [version]**

rpcinfo –b | –d **program version**

There are various options available, and the associated functions are given in the below table:

| Option | Function |
|---|---|
| **-b (broadcast)** | Makes an RPC broadcast to version of program and lists hosts that respond. |
| **-d (delete)** | Removes local RPC registration for version of program. Available to Superuser only. |
| **-n (port number)** | With –t or –u, uses the port numbered port instead of the port number specified by rpcbind. |
| **-p (probe)** | Lists all RPC programs registered with rpcbind on host or on the local system if host is not specified. |
| **-t (TCP)** | Makes a TCP RPC call to version (if specified) of program on host and reports whether it received a response. |
| **-u (UDP)** | Makes a UDP RPC call to version (if specified) of program on host and reports whether it received a response. |

Give the following command to see which RPC programs are registered with the rpcbind daemon (portmapper) on the system named peach:

$ **/usr/sbin/rpcinfo -p peach**

program vers proto port

100000 2 tcp 111 portmapper

100000 2 udp 111 portmapper

100024 1 udp 32768 status

100024 1 tcp 32768 status

100021 1 udp 32769 nlockmgr

100021 3 udp 32769 nlockmgr

### Locking down rpcbind

Because the rpcbind daemon holds information about which servers are running on the local system and which port each server is running on, only trusted systems should have access to this information.

- One way to ensure only selected systems have access to rpcbind is to lock it down in the /etc/hosts.allow and /etc/hosts.deny files.
- Put the following line in hosts.deny preventing all systems from using rpcbind on the local (server) system: rpcbind: ALL

## 10.9 The xinetd Superserver

RHEL uses the xinetd daemon, a more secure replacement for the inetd superserver that was originally shipped with 4.3BSD. Fedora uses the Upstart init daemon for runlevel control and most servers. However, some Fedora servers still require xinetd to be installed and running. The xinetd superserver listens for network connections. When one is made, it launches a specified server daemon and forwards the data from the socket to the daemon's standard input. The version of xinetd distributed with Fedora/RHEL is linked against libwrap.so, so it can use the /etc/hosts.allow and /etc/hosts.deny files for access control. Using TCP wrappers can simplify configuration but hides some of the more advanced features of xinetd. The base configuration for xinetd is stored in the /etc/xinetd.conf file. If this file is not present, xinetd is probably not installed. Working as root, give the following command to install xinetd: # yum install xinetd. The default xinetd.conf file is well commented.

### Securing a Server

You may secure a server either by using TCP wrappers or by setting up a chroot jail.

### TCP Wrappers: Client/Server Security (hosts.allow and hosts.deny)

When you open a local system to access from remote systems, you must ensure that the following criteria are met: Open the local system only to systems you want to allow to access it. Allow each remote system to access only the data you want it to access. Allow each remote system to access data only in the appropriate manner (readonly, read/write, write only). As part of the client/server model, TCP wrappers, which can be used for any daemon that is linked against libwrap.so, rely on the /etc/hosts.allow and /etc/hosts.deny files as the basis of a simple access control language. This access control language defines rules that selectively allow clients to access server daemons on a local system based on the client's address and the daemon the client tries to access.

Each line in the hosts.allow and hosts.deny files has the following format:

daemon_list : client_list [: command]

where daemon_list is a comma-separated list of one or more server daemons (such as rpcbind, vsftpd, or sshd), client_list is a comma-separated list of one or more clients and the optional command is the command that is executed when a client from client_list tries to access a server daemon from daemon_list.

When a client requests a connection with a local server, the hosts.allow and hosts.deny files are consulted in the following manner until a match is found:

1. If the daemon/client pair matches a line in hosts.allow, access is granted.
2. If the daemon/client pair matches a line in hosts.deny, access is denied.

3. If there is no match in either the hosts.allow or hosts.deny files, access is granted.

### Setting Up a chroot Jail

On early UNIX systems, the root directory was a fixed point in the filesystem. On modern UNIX variants, including Linux, you can define the root directory on a preprocess basis. The chroot utility allows you to run a process with a root directory other than **/.** The root directory appears at the top of the directory hierarchy and has no parent: A process cannot access any files above the root directory (because they do not exist). If, for example, you run a program (process) and specify its root directory as /home/sam/jail, the program would have no concept of any files in /home/sam or above: jail is the program's root directory and is labeled / (not jail). By creating an artificial root directory, frequently called a (chroot) jail, you prevent a program from accessing or modifying—possibly maliciously—files outside the directory hierarchy starting at its root. You must set up a chroot jail properly to increase security: If you do not set up the chroot jail correctly, you can make it easier for a malicious user to gain access to a system than if there were no chroot jail.

## 10.10 DHCP: Configures Hosts

Instead of storing network configuration information in local files on each system, DHCP (Dynamic Host Configuration Protocol) enables client systems to retrieve network configuration information each time they connect to the network. A DHCP server assigns IP addresses from a pool of addresses to clients as needed. Assigned addresses are typically temporary but need not be. This technique has several advantages over storing network configuration information in local files:

1) A new user can set up an Internet connection without having to deal with IP addresses, netmasks, DNS addresses, and other technical details. An experienced user can set up a connection more quickly.

2) DHCP facilitates assignment and management of IP addresses and related network information by centralizing the process on a server. A system administrator can configure new systems, including laptops that connect to the network from different locations, to use DHCP; DHCP then assigns IP addresses only when each system connects to the network. The pool of IP addresses is managed as a group on the DHCP server.

3) IP addresses can be used by more than one system, reducing the total number of IP addresses needed. This conservation of addresses is important because the Internet is quickly running out of IPv4 addresses. Although a particular IP address can be used by only one system at a time, many end-user systems require addresses only occasionally, when they connect to the Internet. By reusing IP addresses, DHCP lengthens the life of the IPv4 protocol.

DHCP is particularly useful for administrators who are responsible for maintaining many systems because individual systems no longer need to store unique configuration information.

### How DHCP Works

The client daemon, dhclient (part of the dhcp package), contacts the server daemon, dhcpd, to obtain the IP address, netmask, broadcast address, nameserver address, and other networking parameters. The server provides a lease on the IP address to the client. The client can request the specific terms of the lease, including its duration; the server can, in turn, limit these terms. While connected to the network, a client typically requests extensions of its lease as necessary, so its IP address remains the same. The lease can expire once the client is disconnected from the network, with the server giving the client a new IP address when it requests a new lease. You can also set up a DHCP server to provide static IP addresses for specific clients

### DHCP Client

A DHCP client requests network configuration parameter from the DHCP server and uses those parameters to configure its network interface. The prerequisites is to install the following package: dhclient. When a DHCP client system connects to the network, dhclient requests a lease from the DHCP server and configures the client's network interface(s). Once a DHCP client has requested

and established a lease, it stores information about the lease in a file named dhclient.leases, which is stored in the /var/lib/dhclient directory. This information is used to reestablish a lease when either the server or the client needs to reboot. The DHCP client configuration file, /etc/dhclient.conf, is required only for custom configurations.

### DHCP Server

The DHCP server maintains a list of IP addresses and other configuration parameters. When requested to do so, the DHCP server provides configuration parameters to a client. The prerequisites is to install the following package: **dhcp.** Run chkconfig to cause dhcpd to start when the system enters multiuser mode: # /sbin/chkconfig dhcpd on.

Start dhcpd: # /sbin/service dhcpd start. A simple DHCP server allows you to add clients to a network without maintaining a list of assigned IP addresses. A simple network, such as a home LAN sharing an Internet connection, can use DHCP to assign a dynamic IP address to almost all nodes. The exceptions are servers and routers, which must be at known network locations to be able to receive connections. If servers and routers are configured without DHCP, you can specify a simple DHCP server configuration in /etc/dhcp/dhcpd.conf (FEDORA) or /etc/dhcpd.conf (RHEL):

$ **cat /etc/dhcp/dhcpd.conf**

default-lease-time 600;

max-lease-time 86400;

option subnet-mask 255.255.255.0;

option broadcast-address 192.168.1.255;

option routers 192.168.1.1;

option domain-name-servers 192.168.1.1;

subnet 192.168.1.0 netmask 255.255.255.0 {

range 192.168.1.2 192.168.1.200;

}

Once you have configured a DHCP server, you can start (or restart) it by using the dhcpd init script: # /sbin/service dhcpd restart. Once the server is running, clients configured to obtain an IP address from the server using DHCP should be able to do so.

### Static IP Addresses

Routers and servers typically require static IP addresses. While you can manually configure IP addresses for these systems, it may be more convenient to have the DHCP server provide them with static IP addresses. When a system that requires a specific static IP address connects to the network and contacts the DHCP server, the server needs a way to identify the system so the server can assign the proper IP address to the system. The DHCP server uses the MAC address of the system's Ethernet card (NIC) as an identifier. When you set up the server, you must know the MAC address of each system that requires a static IP address. You can use ifconfig to display the MAC addresses of the Ethernet cards (NICs) in a system. The MAC addresses are the colon-separated series of hexadecimal number pairs following HWaddr:

$ **/sbin/ifconfig | grep -i hwaddr**

eth0 Link encap:Ethernet HWaddr BA:DF:00:DF:C0:FF

eth1 Link encap:Ethernet HWaddr 00:02:B3:41:35:98

## 10.11 Important files and directories in Linux

Filesystems hold directories of files. These structures store user data and system data that are the basis of users' work on the system and the system's existence.

### ~/.bash_profile

It Contains an individual user's login shell initialization script. The shell executes the commands in this file in the same environment as the shell each time a user logs in. The file must be located in a user's home directory. The default Fedora/RHEL .bash_profile file executes the commands in ~/.bashrc. You can use .bash_profile to specify a terminal type (for vi, terminal emulators, and other programs), run stty to establish the terminal characteristics, set up aliases, and perform other housekeeping functions when a user logs in. A simple .bash_profile file specifying a vt100 terminal and CONTROL-H as the erase key follows:

**$ cat .bash_profile**

export TERM=vt100

stty erase '^h'

### ~/.bashrc

It Contains an individual user's interactive, nonlogin shell initialization script. The shell executes the commands in this file in the same environment as the (new) shell each time a user creates a new interactive shell. The .bashrc script differs from .bash_profile in that it is executed each time a new shell is spawned, not just when a user logs in. The default Fedora/RHEL .bash_profile file executes the commands in ~/.bashrc so that these commands are executed when a user logs in.

### /dev

It contains files representing pseudo devices and physical devices that may be attached to the system.

- **/dev/fd0:** The first floppy disk. The second floppy disk is named /dev/fd1.
- **/dev/had:** The master disk on the primary IDE controller. The slave disk on the primary IDE controller is named /dev/hdb. This disk may be a CDROM drive.
- **/dev/hdc:** The master disk on the secondary IDE controller. The slave disk on the secondary IDE controller is named /dev/hdd. This disk may be a CD-ROM drive.
- **/dev/sda:** Traditionally the first SCSI disk; now the first non-IDE drive, including SATA and USB drives. Other, similar drives are named /dev/sdb, /dev/sdc, etc.

These names, such as /dev/sda, represent the order of the devices on the bus the devices are connected to, not the device itself. For example, if you swap the data cables on the disks referred to as /dev/sda and /dev/sdb, the drive's designations will change. Similarly, if you remove the device referred to as /dev/sda, the device that was referred to as /dev/sdb will now be referred to as /dev/sda.

### /dev/disk/by-id

It holds symbolic links to local devices. The names of the devices in this directory identify the devices. Each entry points to the device in /dev that it refers to.

### /dev/disk/by-uuid

It holds symbolic links to local devices. The names of the devices in this directory consist of the UUID numbers of the devices. Each entry points to the device in /dev that it refers to.

### /dev/null

It is also called a bit bucket, output sent to this file disappears. The /dev/null file is a device file and must be created with mknod. Input that you redirect to come from this file appears as nulls, creating an empty file. You can create an empty file named nothing by giving the following command:

**$ cat /dev/null > nothing**

or

**$ cp /dev/null nothing**

or, without explicitly using /dev/null,

**$ > nothing**

### /dev/pts

The /dev/pts pseudofilesystem is a hook into the Linux kernel; it is part of the pseudoterminal support. Pseudoterminals are used by remote login programs, such as ssh and telnet, and xterm as well as by other graphical terminal emulators. The following sequence of commands demonstrates that the user is logged in on /dev/pts/1. After using who am i to verify the line the user is logged in on and using ls to show that this line exists, the user redirects the output of an echo command to /dev/pts/1, whereupon the output appears on the user's screen:

$ **who am i**

alex pts/1 2006-02-16 12:30 (bravo.example.com)

$ **ls /dev/pts**

0 1 2 3 4

$ **echo Hi there > /dev/pts/1**

Hi there

### /dev/random and /dev/urandom

These files are interfaces to the kernel's random number generator. You can use either one with dd to create a file filled with pseudorandom bytes.

$ **dd if=/dev/urandom of=randfile2 bs=1 count=100**

100+0 records in

100+0 records out

100 bytes (100 B) copied, 0.001241 seconds, 80.6 kB/s

The preceding command reads from /dev/urandom and writes to the file named randfile. The block size is 1 and the count is 100 so randfile is 100 bytes long. For bytes that are more random, you can read from /dev/random.

### /dev/zero

Input you take from this file contains an infinite string of zeros (numerical zeros, not ASCII zeros). You can fill a file or overwrite a file with zeros with a command such as the following

$ **dd if=/dev/zero of=zeros bs=1024 count=10**

10+0 records in

10+0 records out

10240 bytes (10 kB) copied, 0.000195 seconds, 52.5 MB/s

$ **od -c zeros**

0000000 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0

*

0024000

### /etc/aliases

It is used by the mail delivery system (typically sendmail) to hold aliases for users. Edit this file to suit local needs.

### /etc/at.allow, /etc/at.deny, /etc/cron.allow, and /etc/cron.deny

By default, users can use the at and cron tab utilities. The at.allow file lists the users who are allowed to use at. The cron.allow file works in the same manner for crontab. The at.deny and cron.deny files specify users who are not permitted to use the corresponding utilities. As Fedora/RHEL is configured, an empty at.deny file and the absence of an at.allow file allows anyone to use at; the absence of cron.allow and cron.deny files allows anyone to use crontab. To prevent anyone except Superuser from using at, remove the at.allow and at.deny files. To prevent anyone except Superuser from using crontab, create a cron.allow file with the single-entry root.

### /etc/dumpdates

It contains information about the last execution of dump. For each filesystem, it stores the time of the last dump at a given dump level. The dump utility uses this information to determine which files to back up when executing at a particular dump level.

### /etc/fstab

Filesystem (mount) table Contains a list of all mountable devices as specified by the system administrator. Programs do not write to this file but only read from it.

### /etc/group

Groups allow users to share files or programs without giving all system users access to those files or programs. This scheme is useful when several users are working with files that are not public. The /etc/group file associates one or more usernames with each group (number). An entry in the /etc/group file has four fields arranged in the following format: **group-name:password:group-ID:login-name-list**

The group-name is the name of the group. The password is an optional encrypted password. This field frequently contains an x, indicating that group passwords are not used. The group-ID is a number, with 1–499 reserved for system accounts. The login-name-list is a comma-separated list of users who belong to the group. The login-name-list is a comma-separated list of users who belong to the group. If an entry is too long to fit on one line, end the line with a backslash (\), which quotes the following RETURN, and continue the entry on the next line.

### /etc/hosts

The /etc/hosts file stores the name, IP address, and optional aliases of the other systems that the local system knows about. At the very least, this file must have the hostname and IP address that you have chosen for the local system and a special entry for localhost. This entry supports the loopback service, which allows the local system to talk to itself (for example, for RPC services). The IP address of the loopback service is always 127.0.0.1. Following is a simple /etc/hosts file for the system named rose with an IP address of 192.168.0.10:

$ **cat /etc/hosts**

# Do not remove the following line, or various programs

# that require network functionality will fail.

127.0.0.1 rose localhost.localdomain localhost

192.168.0.1 bravo.example.com bravo

192.168.0.4 mp3server

192.168.0.5 workstation

192.168.0.10 rose

...

### /etc/inittab (RHEL)

Initialization table Under RHEL, this file controls how the System V init process behaves. Fedora has replace the System V init daemon with the Upstart init daemon. Each line in inittab contains four colon-separated fields: **id:runlevel:action:process.** The id uniquely identifies an entry in the inittab file. The runlevel is the system runlevel(s) at which process is executed. The runlevel consists of zero or more characters chosen from 0123456S. If more than one runlevel is listed, the associated process is executed at each of the specified runlevels. The action is one of the following keywords: respawn, wait, once, boot, bootwait, ondemand, powerfail, powerwait, powerokwait, powerfailnow, ctrlaltdel, kbrequest, off, ondemand, initdefault, or sysinit. The wait keyword instructs init to start the process and wait for it to terminate.

### /etc/motd

It contains the message of the day, which can be displayed each time someone logs in using a textual login. This file typically contains site policy and legal information. Keep this file short because users tend to see the message many times.

### /etc/mtab

When you call mount without any arguments, it consults this file and displays a list of mounted devices. Each time you (or an init script) call mount or umount, these utilities make the necessary changes to mtab. Although this is an ASCII text file, you should not edit it.

### /etc/netgroup

It defines netgroups, which are used for checking permissions when performing remote logins and remote mounts and when starting remote shells.

### /etc/nsswitch.conf

It specifies whether a system uses as the source of certain information NIS, DNS, local files, or a combination, and in what order it consults these services.

### /etc/pam.d

Files in this directory specify the authentication methods used by PAM applications.

### /etc/passwd

It describes users to the system. Do not edit this file directly. Each line in passwd has seven colon-separated fields that describe one user: **login-name:dummy-password:user-ID:group-ID:info: directory: program**

The login-name is the user's username—the name you enter in response to the login: prompt or GUI login screen. The value of the dummy-password is the character x. An encrypted/hashed password is stored in /etc/shadow. For security reasons, every account should have a password. By convention, disabled accounts have an asterisk (*) in this field. The user-ID is a number, with 0 indicating Superuser and 1–499 being reserved for system accounts. The group-ID identifies the user as a member of a group. It is a number, with 0–499 being reserved for system accounts; see /etc/group. You can change these values and set maximum values in /etc/login.defs. The info is information that various programs, such as accounting programs and email, use to identify the user further. Normally it contains at least the first and last names of the user. It is referred to as the GECOS field. The directory is the absolute pathname of the user's home directory. The program is the program that runs once the user logs in. If program is not present, a value of /bin/bash is assumed. You can put /bin/tcsh here to log in using the TC Shell or /bin/zsh to log in using the Z Shell, assuming the shell you specify is installed.

### /etc/printcap

The printer capability database. This file describes system printers and is derived from 4.3BSD UNIX.

### /etc/profile

It contains a systemwide interactive shell initialization script for environment and start-up programs. When you log in, the shell immediately executes the commands in this file in the same environment as the shell. Following is an example of a /etc/profile file that displays the message of the day (the /etc/motd file), sets the file-creation mask, and sets the interrupt character to CONTROL-C:

**# cat /etc/profile**

cat /etc/motd

umask 022

stty intr '^c'

### /etc/protocols

It provides protocol numbers, aliases, and brief definitions for DARPA Internet TCP/IP protocols. Do not modify this file.

### /etc/rc.d

It holds the system init scripts, also called run command (rc) scripts. The init program executes several init scripts each time the system changes state or runlevel.

### /etc/resolv.conf

The resolver configuration file, used to provide access to DNS. The following example shows the resolv.conf file for the example.com domain. A resolv.conf file usually has at least two lines—a domain line and a nameserver line: **# cat /etc/resolv.conf**

domain example.com

nameserver 10.0.0.50

nameserver 10.0.0.51

### /etc/rpc

It maps RPC services to RPC numbers. The three columns in this file show the name of the server for the RPC program, the RPC program number, and any aliases.

### /etc/services

It lists system services. The three columns in this file show the informal name of the service, the port number/protocol the service frequently uses, and any aliases for the service. This file does not specify which services are running on the local system, nor does it map services to port numbers. The services file is used internally to map port numbers to services for display purposes.

### /etc/shadow

It contains encrypted or MD5 hashed user passwords. Each entry occupies one line composed of nine fields, separated by colons: **login-name: password: last-mod: min: max: warn: inactive: expire: flag.** The login-name is the user's username—the name that the user enters in response to the login: prompt or GUI login screen. The password is an encrypted or hashed. The last-mod field

indicates when the password was last modified. The min is the minimum number of days that must elapse before the password can be changed. The max is the maximum number of days before the password must be changed. The warn specifies how much advance warning (in days) to give the user before the password expires. The account will be closed if the number of days between login sessions exceeds the number of days specified in the inactive field. The account will also be closed as of the date in the expire field. The last field in an entry, flag, is reserved for future use. You can use the Password Info tab in system-config-users to modify these fields.

### /etc/sysconfig

A directory containing a hierarchy of system configuration files.

### /proc

The /proc pseudofilesystem provides a window into the Linux kernel. Through /proc you can obtain information on any process running on your computer, including its current state, memory usage, CPU usage, terminal, parent, and group. You can extract information directly from the files in /proc.

### /sbin/shutdown

A utility that brings the system down.

### swap

Even though swap is not a file, swap space can be added and deleted from the system dynamically. Swap space is used by the virtual memory subsystem. When it runs low on real memory (RAM), the system writes memory pages from RAM to the swap space on the disk. Which pages are written and when they are written are controlled by finely tuned algorithms in the Linux kernel. When needed by running programs, these pages are brought back into RAM—a technique called paging. When a system is running very short on memory, an entire process may be paged out to disk.

### /sys

The /sys pseudofilesystem was added in the Linux 2.6 kernel to make it easy for programs running in kernelspace, such as device drivers, to exchange information with programs running in userspace.

### /usr/share/magic

Most files begin with a unique identifier called a magic number. This file is a text database listing all known magic numbers on the system. When you use the file utility, it consults /usr/share/magic to determine the type of a file. Occasionally you may acquire a new tool that creates a new type of file that is unrecognized by the file utility. In this situation you need to update the /usr/share/magic file

### /var/log

It holds system log files.

### /var/log/messages

It contains messages from daemons, the Linux kernel, and security programs. For example, you will find filesystem full warning messages, error messages from system daemons (NFS, rsyslog, printer daemons), SCSI and IDE disk error messages, and more in messages. Check /var/log/messages periodically to keep informed about important system events. Much of the information displayed

on the system console is also sent to messages. If the system experiences a problem and you cannot access the console, check this file for messages about the problem.

### /var/log/secure

It holds messages from security-related programs such as su and the sshd daemon.

## 10.12 Types of files

Linux supports many types of files:

1) Ordinary files, directories, links, and inodes.

2) Symbolic links.

3) Special files.

4) FIFO special file.

5) Sockets.

6) Block and character devices.

7) Raw devices.

### Ordinary Files, Directories, Links, and Inodes

#### Ordinary and directory files

An ordinary file stores user data, such as textual information, programs, or images, such as a jpeg or tiff file. A directory is a standard-format disk file that stores information, including names, about ordinary files, and other directory files.

#### Inodes

An inode is a data structure, stored on disk, that defines a file's existence and is identified by an inode number. An inode contains critical information, such as the name of the owner of the file, where it is physically located on the disk, and how many hard links point to it. In addition, SELinux stores extended information about files in inodes. A directory relates each of the filenames it stores to an inode. When you move (mv) a file within a filesystem, you change the filename portion of the directory entry associated with the inode that describes the file. You do not create a new inode.

If you move a file to another filesystem, mv first creates a new inode on the destination filesystem and then deletes the original inode. You can also use mv to move a directory recursively, in which case all files in the directory are copied and deleted. When you remove (rm) a file, you delete the directory entry that describes the file. When you remove the last hard link to a file, the operating system puts all blocks the inode pointed to back in the free list (the list of blocks that are available for use on the disk) and frees the inode to be used again.

#### The . and .. directory entries

Every directory contains at least two entries (. and ..). The . entry is a link to the directory itself. The .. entry is a link to the parent directory. In the case of the root directory, there is no parent and the .. entry is a link to the root directory itself. It is not possible to create hard links to directories.

### Symbolic links

Because each filesystem has a separate set of inodes, you can create hard links to a file only from within the filesystem that holds that file. To get around this limitation, Linux provides symbolic links, which are files that point to other files. Files that are linked by a symbolic link do not share an inode. Therefore, you can create a symbolic link to a file from any filesystem. You can also create a symbolic link to a directory, device, or other special file.

## Special Files

Special files represent Linux kernel routines that provide access to an operating system feature. FIFO (first in, first out) special files allow unrelated programs to exchange information. Sockets allow unrelated processes on the same or different computers to exchange information. One type of socket, the UNIX domain socket, is a special file. Symbolic links are another type of special file.

### Device files

Device files, which include both block and character special files, represent device drivers that let you communicate with peripheral devices, such as terminals, printers, and hard disks. By convention, device files appear in the /dev directory and its subdirectories. Each device file represents a device: You read from and write to the file to read from and write to the device it represents. For example, using cat to send an audio file to /dev/dsp plays the file. The following example shows part of the output that an ls –l command produces for the /dev directory:

$ **ls -l /dev**

total 0

crw-rw---- 1 root root 14, 12 Jan 25 08:33 adsp

crw------- 1 root root 10, 175 Jan 25 08:33 agpgart

crw------- 1 zach root 14, 4 Jan 25 08:33 audio

drwxr-xr-x 3 root root 60 Jan 25 08:33 bus

lrwxrwxrwx 1 root root 3 Jan 25 08:33 cdrom -> hdb

lrwxrwxrwx 1 root root 3 Jan 25 08:33 cdwriter -> hdb

.....

The first character of each line is always –, b, c, d, l, or p, representing ordinary (plain), block, character, directory, symbolic link, or named pipe (see the following section), respectively. The next nine characters identify the permissions for the file, followed by the number of hard links and the names of the owner and group. Where the number of bytes in a file would appear for an ordinary or directory file, a device file shows major and minor device numbers separated by a comma. The rest of the line is the same as any other ls –l listing.

### udev

The udev utility manages device naming dynamically. It replaces the earlier devfs and moves the device naming functionality from the kernel to userspace. Because devices are added to and removed from a system infrequently, the performance penalty associated with this change is minimal. The benefit of the move is that a bug in udev cannot compromise or crash the kernel. The udev utility is part of the hotplug system. When a device is added to or removed from the system, the kernel creates a device name in the /sys pseudofilesystem and notifies hotplug of the event, which is received by udev. The udev utility then creates the device file, usually in the /dev directory, or removes the device file from the system. The udev utility can also rename network interfaces.

### Hotplug

The hotplug system allows you to plug a device into the system and use it immediately. Although hotplug was available in the Linux 2.4 kernel, the 2.6 kernel integrates hotplug with the unified device driver model framework.

## FIFO Special Files (Named Pipe)

A FIFO special file, also called a named pipe, represents a pipe: You read from and write to the file to read from and write to the pipe. The term FIFO stands for first in, first out—the way any pipe

works. In other words, the first information that you put in one end is the first information that comes out the other end. When you use a pipe on a command line to send the output of a program to the printer, the printer outputs the information in the same order that the program produced it and sent it to the pipe. The UNIX and Linux systems have included pipes for many generations. Without named pipes, only processes that were children of the same ancestor could use pipes to exchange information. Using named pipes, any two processes on a single system can exchange information. When one program writes to a FIFO special file, another program can read from the same file. The programs do not have to run at the same time or be aware of each other's activity. The operating system handles all buffering and information storage. This type of communication is termed asynchronous (async) because programs on the ends of the pipe do not have to be synchronized.

## Sockets

Like a FIFO special file, a socket allows asynchronous processes that are not children of the same ancestor to exchange information. Sockets are the central mechanism of the interprocess communication that forms the basis of the networking facility. When you use networking utilities, pairs of cooperating sockets manage the communication between the processes on the local computer and the remote computer. Sockets form the basis of such utilities as ssh and scp.

### Major and Minor Device Numbers

A major device number points to a driver in the kernel that works with a class of hardware devices: terminal, printer, tape drive, hard disk, and so on. In the list of the /dev directory, all of the hard disk partitions have a major device number of 3. A minor device number represents a particular piece of hardware within a class. Although all hard disk partitions are grouped together by their major device number, each has a different minor device number (sda1 is 1, sda2 is 2, and so on). This setup allows one piece of software (the device driver) to service all similar hardware yet to be able to distinguish among different physical units.

## Block and Character Devices

A block device is an I/O (input/output) device that is characterized by

- Being able to perform random access reads.
- Having a specific block size.
- Handling only single blocks of data at a time.
- Accepting only transactions that involve whole blocks of data.
- Being able to have a filesystem mounted on it.
- Having the Linux kernel buffer its input and output.

Appearing to the operating system as a series of blocks numbered from 0 through n – 1, where n is the number of blocks on the device. Block devices commonly found on a Linux system include hard disks, floppy diskettes, and CDs.

A character device is any device that is not a block device. Examples of character devices include printers, terminals, tape drives, and modems. The device driver for a character device determines how a program reads from and writes to that device. For example, the device driver for a terminal allows a program to read the information you type on the terminal in two ways:

- First, a program can read single characters from a terminal in raw mode—that is, without the driver doing any interpretation of the characters.
- Alternatively, a program can read one line at a time. When a program reads one line at a time, the driver handles the erase and kill characters, so the program never sees typing mistakes that have been corrected. In this case, the program reads everything from the beginning of a line to the RETURN that ends a line; the number of characters in a line can vary.

### Raw Devices

Device driver programs for block devices usually have two entry points so they can be used in two ways: as block devices or as character devices. The character device form of a block device is called a raw device. A raw device is characterized by

- Direct I/O (no buffering through the Linux kernel).
- A one-to-one correspondence between system calls and hardware requests.
- Device-dependent restrictions on I/O.

An example of a utility that uses a raw device is fsck. It is more efficient for fsck to operate on the disk as a raw device, rather than being restricted by the fixed size of blocks in the block device interface. Because it has full knowledge of the underlying filesystem structure, fsck can operate on the raw device using the largest possible units. When a filesystem is mounted, processes normally access the disk through the block device interface, which explains why it is important to allow fsck to modify only an unmounted filesystem. On a mounted filesystem, there is the danger that, while fsck is rearranging the underlying structure through the raw device, another process could change a disk block using the block device, resulting in a corrupted filesystem.

## 10.13 Filesystems

| Filesystem | Features |
|---|---|
| **adfs** | Advanced Disc Filing System. Used on Acorn computers. The word Advanced differentiated this filesystem from its predecessor, DFS, which did not support advanced features such as a hierarchical filesystem. |
| **affs** | Amiga Fast Filesystem (FFS). |
| **autofs** | Automounting filesystem |
| **coda** | CODA distributed filesystem |
| **devpts** | A pseudofilesystem for pseudoterminals |
| **ext2** | A standard filesystem for Fedora/RHEL systems, usually with the ext3 extension. |
| **ext3** | A journaling extension to the ext2 filesystem. It greatly improves recovery time from crashes (it takes a lot less time to run fsck), promoting increased availability. As with any filesystem, a journaling filesystem can lose data during a system crash or hardware failure. |
| **ext4** | An extension of the ext3 filesystem |
| **GFS** | Global Filesystem. GFS is a journaling, clustering filesystem. It enables a cluster of Linux servers to share a common storage pool. |
| **hfs** | Hierarchical Filesystem: used by older Macintosh systems. Newer Macintosh |

| | |
|---|---|
| | systems use hfs+. |
| **hpfs** | High-Performance Filesystem: the native filesystem for IBM's OS/2. |
| **iso9660** | The standard filesystem for CD-ROMs. |
| **minix** | Very similar to Linux, the filesystem of a small operating system that was written for educational purposes by Andrew S. Tanenbaum |
| **msdos** | The filesystem used by DOS and subsequent Microsoft operating systems. Do not use msdos for mounting Windows filesystems; it does not read VFAT attributes. |
| **ncpfs** | Novell NetWare NCP Protocol Filesystem: used to mount remote filesystems under NetWare. |
| **nfs** | Network Filesystem. Developed by Sun Microsystems, this protocol allows a computer to access remote files over a network as if they were local |
| **ntfs** | NT Filesystem: the native filesystem of Windows NT. |
| **proc** | An interface to several Linux kernel *data structures that behaves* like a filesystem |
| **qnx4** | QNX 4 operating system filesystem |
| **reiserfs** | A journaling filesystem, based on balanced-tree algorithms. |
| **romfs** | A dumb, readonly filesystem used mainly for *RAM disks during* installation |
| **smbfs** | Samba Filesystem |
| **software RAID** | RAID implemented in software. |
| **sysv** | System V UNIX filesystem. |
| **ufs** | Default filesystem under Sun's Solaris operating system and other UNIXs. |
| **umsdos** | A full-feature UNIX-like filesystem that runs on top of a DOS FAT filesystem. |
| **vfat** | Developed by Microsoft, a standard that allows long filenames on FAT partitions. |
| **xfs** | SGI's journaled filesystem (ported from Irix). |

## 10.14 mount: Mounts a Filesystem

The mount utility connects directory hierarchies—typically filesystems—to the Linux directory hierarchy. These directory hierarchies can be on remote and local disks, CDs, and floppy diskettes. Linux also allows you to mount virtual filesystems that have been built inside ordinary files, filesystems built for other operating systems, and the special /proc filesystem, which maps useful Linux kernel information to a pseudodirectory.

**Mount point:**

The mount point for the filesystem/directory hierarchy that you are mounting is a directory in the local filesystem. This directory must exist before you can mount a filesystem; its contents disappear as long as a filesystem is mounted on it and reappear when you unmount the filesystem. Without any arguments, mount lists the currently mounted filesystems, showing the physical device holding each filesystem, the mount point, the type of filesystem, and any options set when each filesystem was mounted:

```
$ mount
proc on /proc type proc (rw)
/dev/hdb1 on / type ext2 (rw)
/dev/hdb4 on /tmp type ext2 (rw)
/dev/hda5 on /usr type ext3 (rw)
/dev/sda1 on /usr/X386 type ext2 (rw)
/dev/sda3 on /usr/local type ext2 (rw)
/dev/hdb3 on /home type ext3 (rw)
/dev/hda1 on /dos type msdos (rw,umask=000)
tuna:/p04 on /p04 type nfs (rw,addr=192.168.0.8)
/dev/scd0 on /mnt/cdrom type iso9660 (ro,noexec,nosuid,nodev)
```

The mount utility gets this information from the /etc/mtab file. The first entry in the preceding example shows the /proc pseudofilesystem. The next six entries identify disk partitions holding standard Linux ext2 and ext3 filesystems. These partitions are found on three disks: two IDE disks (hda, hdb) and one SCSI disk (sda). Disk partition /dev/hda1 has a DOS (msdos) filesystem mounted at the directory /dos in the Linux filesystem. You can access the DOS files and directories on this partition as if they were Linux files and directories, using Linux utilities and applications. The line starting with tuna shows a mounted, remote NFS filesystem. The last line shows a CD mounted on a SCSI CD drive (/dev/scd0).

### Do not mount anything on root (/):

Always mount network directory hierarchies and removable devices at least one level below the root level of the filesystem. The root filesystem is mounted on /; you cannot mount two filesystems in the same place. If you were to try to mount something on /, all files, directories, and filesystems that were under the root directory would no longer be available, and the system would crash.

### Mount Options

The mount utility takes many options, which you can specify on the command line or in the /etc/fstab file. For a complete list of mount options for local filesystems, see the mount man page; for remote directory hierarchies, see the nfs man page. The **noauto** option causes Linux not to mount the filesystem automatically. The **nosuid** option forces mounted setuid executables to run with regular permissions (no effective user ID change) on the local system (the system that mounted the filesystem). Unless you specify the user, users, or owner option, only Superuser can mount and unmount a filesystem. The user option means that any user can mount the filesystem, but the filesystem can be unmounted only by the same user who mounted it; users means that any

user can mount and unmount the filesystem. These options are frequently specified for CD and floppy drives. The owner option, which is used only under special circumstances, is similar to the user option except that the user mounting the device must own the device.

## Mounting a Linux Floppy Diskette

Mounting a Linux floppy diskette is similar to mounting a partition of a hard disk. Put an entry similar to the following in /etc/fstab for a diskette in the first floppy drive:

**/dev/fd0 /mnt/floppy auto noauto,users 0 0**

Specifying a filesystem type of auto causes the system to probe the filesystem to determine its type and allows users to mount a variety of diskettes. Create the /mnt/floppy directory if necessary. Insert a diskette and try to mount it. The diskette must be formatted (use fdformat). Use mkfs to create a filesystem—but be careful, because mkfs destroys all data on the diskette: **# mount /dev/fd0**

mount: you must specify the filesystem type

**# mkfs /dev/fd0**

mke2fs 1.41.9 (22-Aug-2009)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

184 inodes, 1440 blocks

72 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=1572864

1 block group

8192 blocks per group, 8192 fragments per group

184 inodes per group

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

Try the mount command again:

**# mount /dev/fd0**

**# mount**

...

/dev/fd0 on /mnt/floppy type ext2 (rw,noexec,nosuid,nodev)

**# df -h /dev/fd0**

Filesystem Size Used Avail Use% Mounted on

/dev/fd0 1.4M 19K 1.3M 2% /mnt/floppy

The mount command without any arguments and df –h /dev/fd0 show the floppy is mounted and ready for use.

## umount: Unmounts a Filesystem

The **umount** utility unmounts a filesystem as long as it does not contain any files or directories that are in use (open). For example, a logged-in user's working directory must not be on the filesystem you want to unmount. The next command unmounts the CD mounted earlier: $ **umount /mnt/cdrom.** Unmount a floppy or a remote directory hierarchy the same way you would unmount a partition of a hard drive. The umount utility consults /etc/fstab to get the necessary information and then unmounts the appropriate filesystem from its server. When a process has a file open on the filesystem that you are trying to unmount, umount displays a message similar to the following:

umount: /home: device is busy

Use the –a option to umount to unmount all filesystems, except for the one mounted at /, which can never be unmounted. You can combine –a with the –t option to unmount filesystems of a given type (ext3, nfs, or others). For example, the following command unmounts all mounted nfs directory hierarchies that are not being used: # umount -at nfs

## fstab: Keeps Track of Filesystems

The system administrator maintains the /etc/fstab file, which lists local and remote directory hierarchies, most of which the system mounts automatically when it boots. The fstab file has six columns, where a hyphen is a placeholder for a column that has no value:

1) Name—The name, label, or UUID number of a local block device or a pointer to a remote directory hierarchy. When you install the system, Fedora/RHEL uses UUID numbers for fixed devices. It prefaces each line in fstab that specifies a UUID with a comment that specifies the device name. Using UUID numbers in fstab during installation circumvents the need for consistent device naming.

2) Mount point—The name of the directory file that the filesystem/directory hierarchy is to be mounted on. If it does not already exist, create this directory using mkdir.

3) Type—The type of filesystem/directory hierarchy that is to be mounted. Local filesystems are generally of type ext2, ext3, or iso9660, and remote directory hierarchies are of type nfs or cifs.

4) Mount options—A comma-separated list of mount options, such as whether the filesystem is mounted for reading and writing (rw, the default) or readonly (ro).

5) Dump—Used by dump to determine when to back up the filesystem.

6) Fsck—Specifies the order in which fsck checks filesystems. Root (/) should have a 1 in this column. Filesystems that are mounted to a directory just below the root directory should have a 2. Filesystems that are mounted on another mounted filesystem (other than root) should have a 3.

The following example shows a typical **fstab file:**

**# cat /etc/fstab**

LABEL=/1 / ext3 defaults 1 1

LABEL=/boot1 /boot ext3 defaults 1 2

devpts /dev/pts devpts gid=5,mode=620 0 0

tmpfs /dev/shm tmpfs defaults 0 0

LABEL=/home1 /home ext3 defaults 1 2

proc /proc proc defaults 0 0

sysfs /sys sysfs defaults 0 0

LABEL=SWAP-hda5 swap swap defaults 0 0

/dev/hda3 /oldhome ext3 defaults 0 0

tuna:/p04 /p04 nfs defaults 0 0

/dev/fd0 /mnt/floppy auto noauto,users 0 0

## fsck: Checks Filesystem Integrity

The **fsck (filesystem check)** utility verifies the integrity of filesystems and, if possible, repairs any problems it finds. Because many filesystem repairs can destroy data, particularly on a nonjournaling filesystem, such as ext2, by default fsck asks you for confirmation before making each repair. The following command checks all unmounted filesystems that are marked to be checked in /etc/fstab except for the root filesystem: **# fsck -AR**

The –A option causes fsck to check filesystems listed in fstab. The –R option checks the same filesystems as –A except it does not check the root filesystem. You can check a specific filesystem with a command similar to one of the following:

**# fsck /home**

or

**# fsck /dev/sda6**

**Crash flag:** The /etc/rc.d/rc.sysinit start-up script looks for two flags in the root directory of each partition to determine whether fsck needs to be run on that partition before it is mounted. The .autofsck flag (the crash flag) indicates that the partition should be checked.

## tune2fs: Changes Filesystem Parameters

The tune2fs utility displays and modifies filesystem parameters on ext2, ext3, and ext4 filesystems. This utility can also set up journaling on an ext2 filesystem, turning it into an ext3 filesystem. With the introduction of increasingly more reliable hardware and software, systems are rebooted less frequently, so it is important to check filesystems regularly. By default, fsck is run on each partition while the system is brought up, before the partition is mounted. Depending on the flags, fsck may do nothing more than display a message saying that the filesystem is clean. The larger the partition, the more time it takes to check it, assuming a nonjournaling filesystem. These checks are often unnecessary. The tune2fs utility helps you to find a happy medium between checking filesystems each time you reboot the system and never checking them. It does so by scheduling when fsck checks a filesystem (these checks occur only when the system is booted). You can use two scheduling patterns: time elapsed since the last check and number of mounts since the last check. The following command causes /dev/sda6 to be checked when fsck runs after it has been mounted eight times or after 15 days have elapsed since its last check, whichever happens first:

**# tune2fs -c 8 -i 15 /dev/sda6**

tune2fs 1.41.9 (22-Aug-2009)

Setting maximal mount count to 8

Setting interval between checks to 1296000 seconds

The next tune2fs command is similar but works on a different partition and sets the current mount count to 4. When you do not specify a current mount count, it is set to zero:

**# tune2fs -c 8 -i 15 -C 4 /dev/sda6**

tune2fs 1.41.9 (22-Aug-2009)

Setting maximal mount count to 8

Setting current mount count to 4

Setting interval between checks to 1296000 seconds

RAID Filesystem

RAID (Redundant Arrays of Inexpensive/Independent Disks) spreads information across several disks to combine several physical disks into one larger virtual device. RAID improves performance and creates redundancy. More than six types of RAID configurations exist. Using Fedora/RHEL

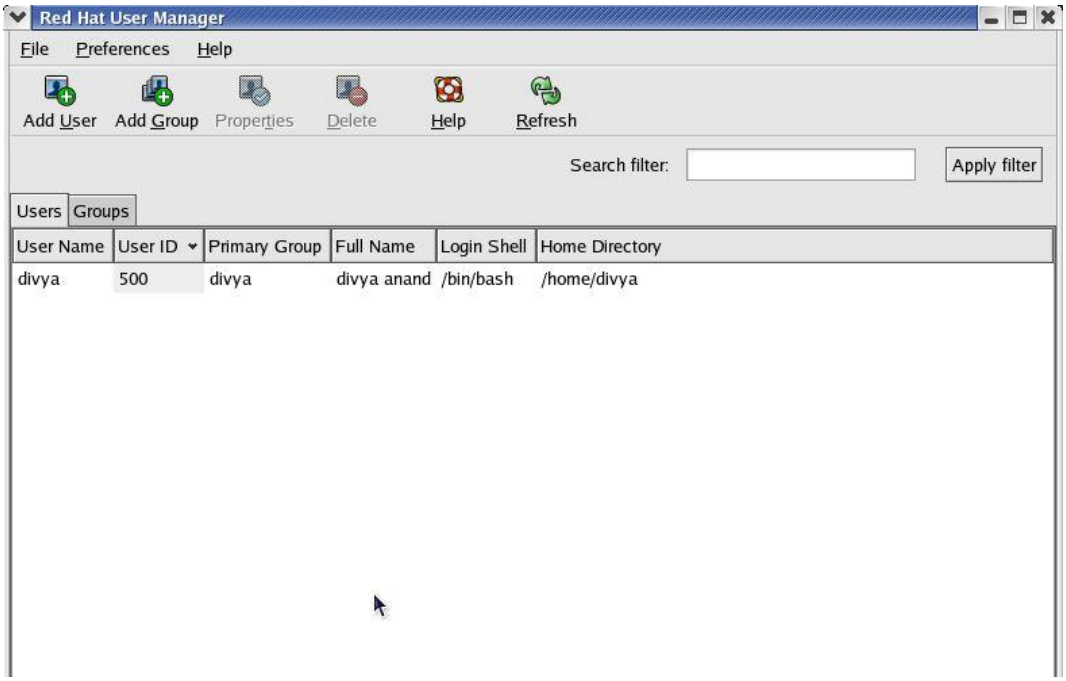
tools, you can set up software RAID. Hardware RAID requires hardware that is designed to implement RAID. RAID can be an effective addition to a backup. Fedora/RHEL offers RAID software that you can install either when you install a Fedora/RHEL system or as an afterthought. The Linux kernel can automatically detect RAID disk partitions at boot time if the partition ID is set to 0xfd, which fdisk recognizes as Linux **raid autodetect.** Software RAID, as implemented in the kernel, is much cheaper than hardware RAID. Not only does this software avoid specialized RAID disk controllers, but it also works with the less expensive IDE disks as well as SCSI disks.

## 10.15 Configuring User and Group Accounts

More than a username is required for a user to be able to log in and use a system. At a minimum a user must have an entry in the **/etc/passwd** and **/etc/shadow** files and a **home directory.**

### system-config-users: Manages User Accounts

The **system-config-users** utility displays the User Manager window and enables you to add, delete, and modify system users and groups. To display the User Manager window, enter **system-config-users** on a command line or select **Main menu: System | Administration | Users and Groups**.




This window has two tabs: **Users and Groups**, where each tab displays information appropriate to its name.

#### Search filter

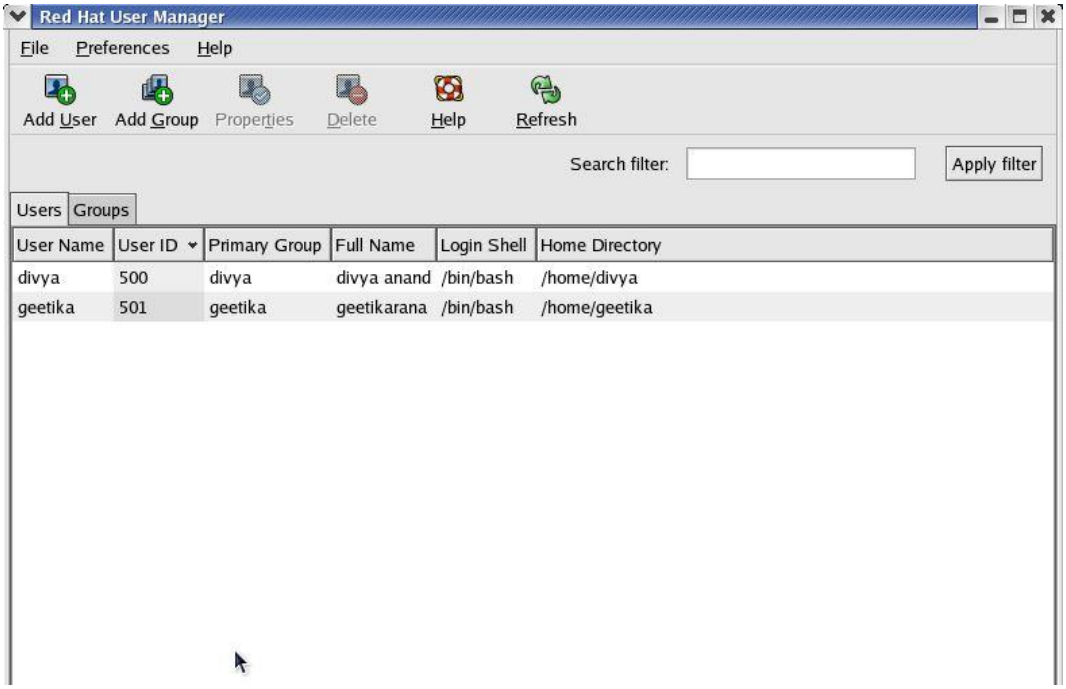
The **Search filter**, located just below the toolbar, selects users or groups whose names match the string, which can include wildcards, that you enter in the Search filter text box. The string matches the beginning of a name. For example, *nob matches nobody and nfsnobody, whereas nob matches only nobody.

#### Adding a user

To create a new user, click the **Add User button** on the toolbar. The User Manager displays the Create New User window. Enter the information for the new user and click OK.

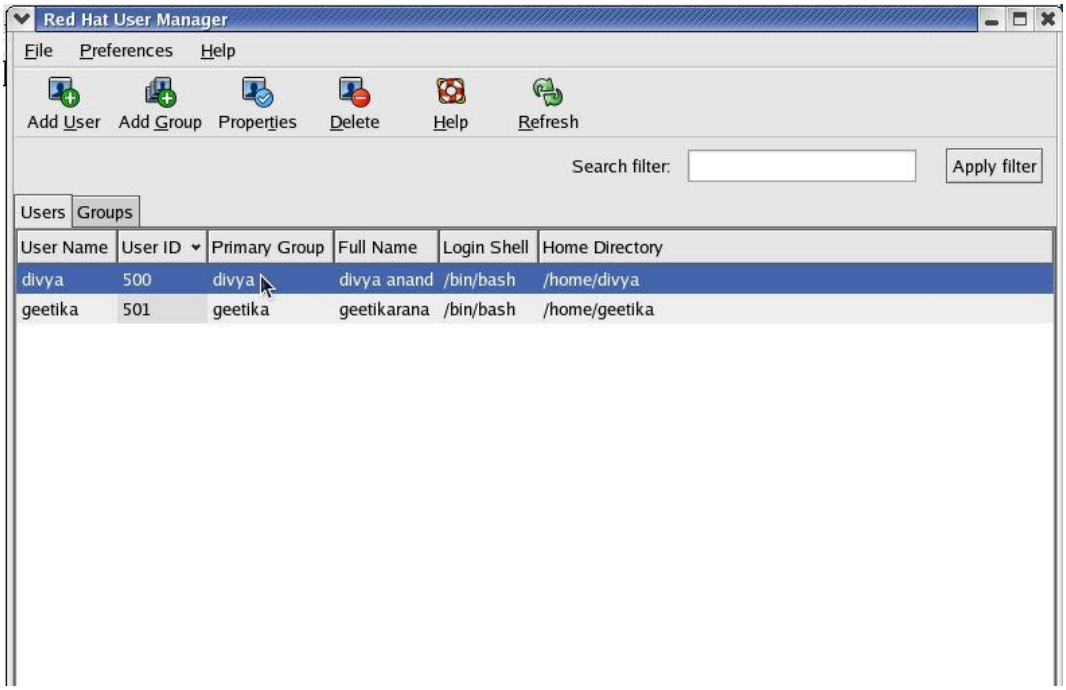

Create New User
User Name:
Full Name:
Password:
Confirm Password:
Login Shell:
/bin/bash
Create home directory
Home Directory:
Create a private group for the user
Specify user ID manually
UID:
500
Cancel
OK

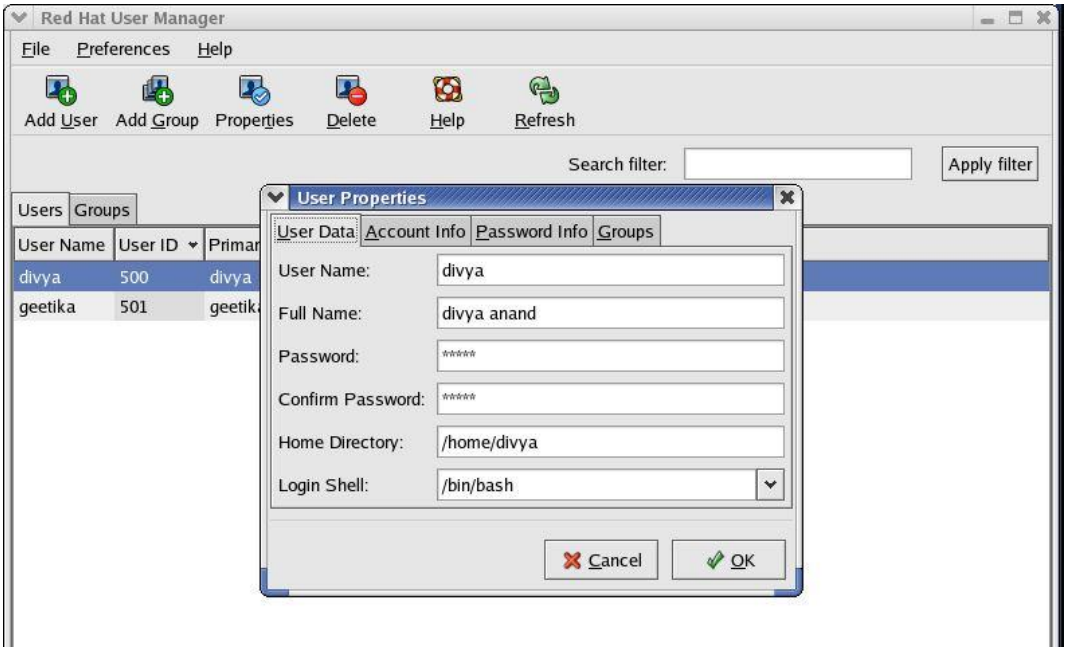

Create New User
User Name:
geetika
Full Name:
geetikarana
Password:
Confirm Password:
Login Shell:
/bin/bash
Create home directory
Home Directory:
/home/geetika
Create a private group for the user
Specify user ID manually
UID:
500
Cancel
OK




## Modifying a user

Once you create a user, you can **modify the user** to add/change/remove information. To modify a user, highlight the user in the User Manager window and click Properties on the toolbar; the utility displays the User Properties window.



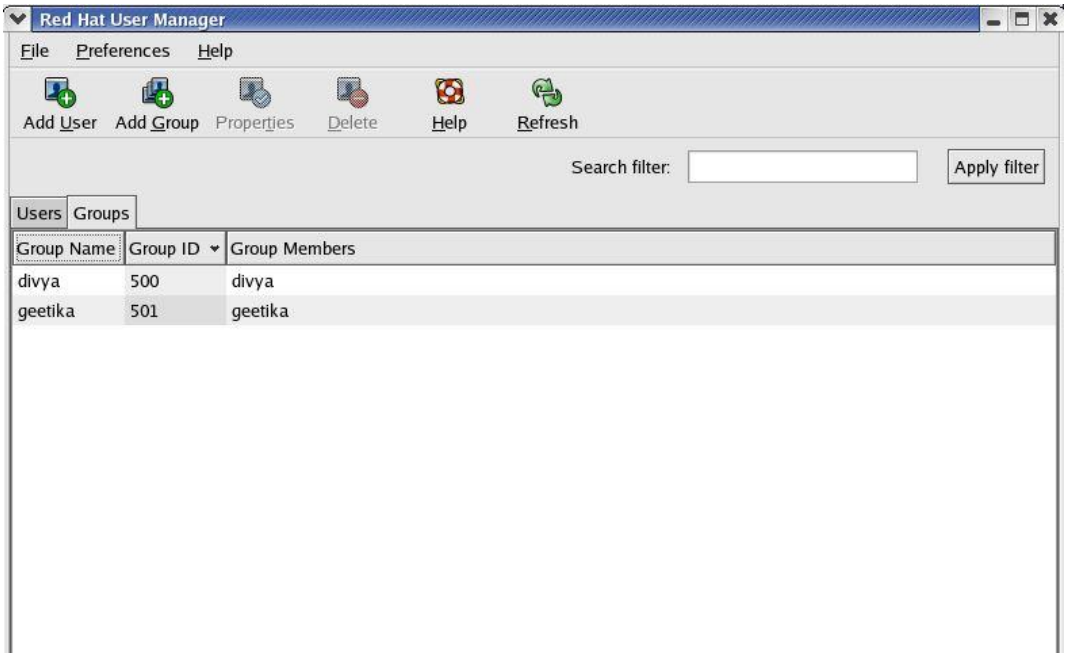



The User Properties window has four tabs: User Data, Account Info, Password Info, and Groups.

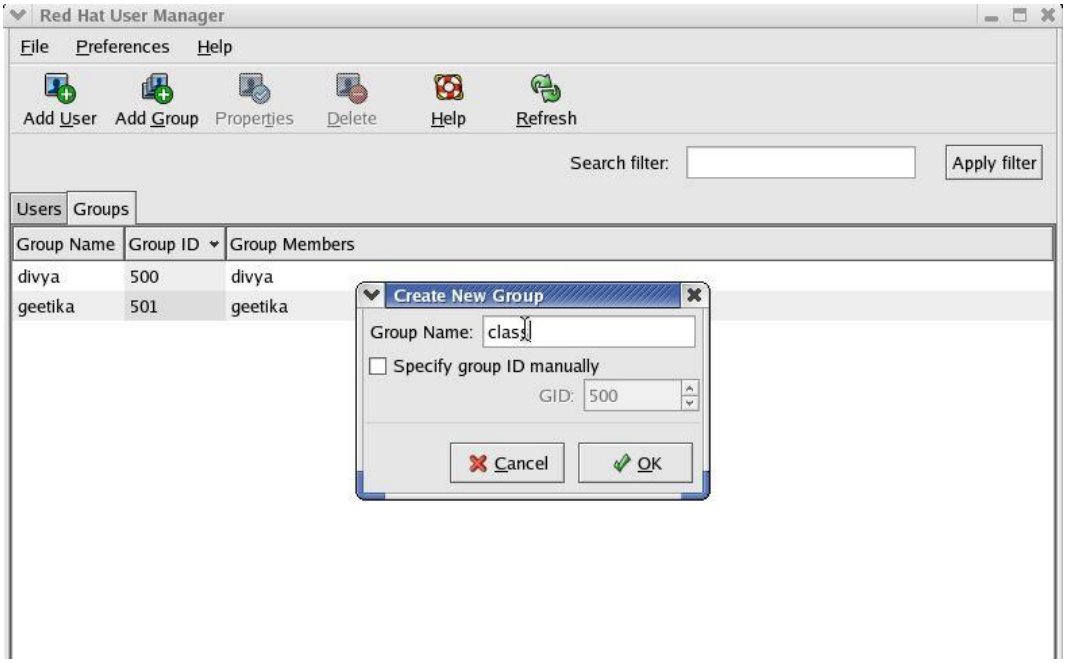
- **User Data tab**: It holds basic user information such as name and password.
- **Account Info tab**: It allows you to specify an expiration date for the account and to lock the account so the user cannot log in.
- **Password Info tab**: It allows you to turn on password expiration and specify various related parameters.
- **Groups tab**: You can specify the groups that the user is a member of.

## Working with groups

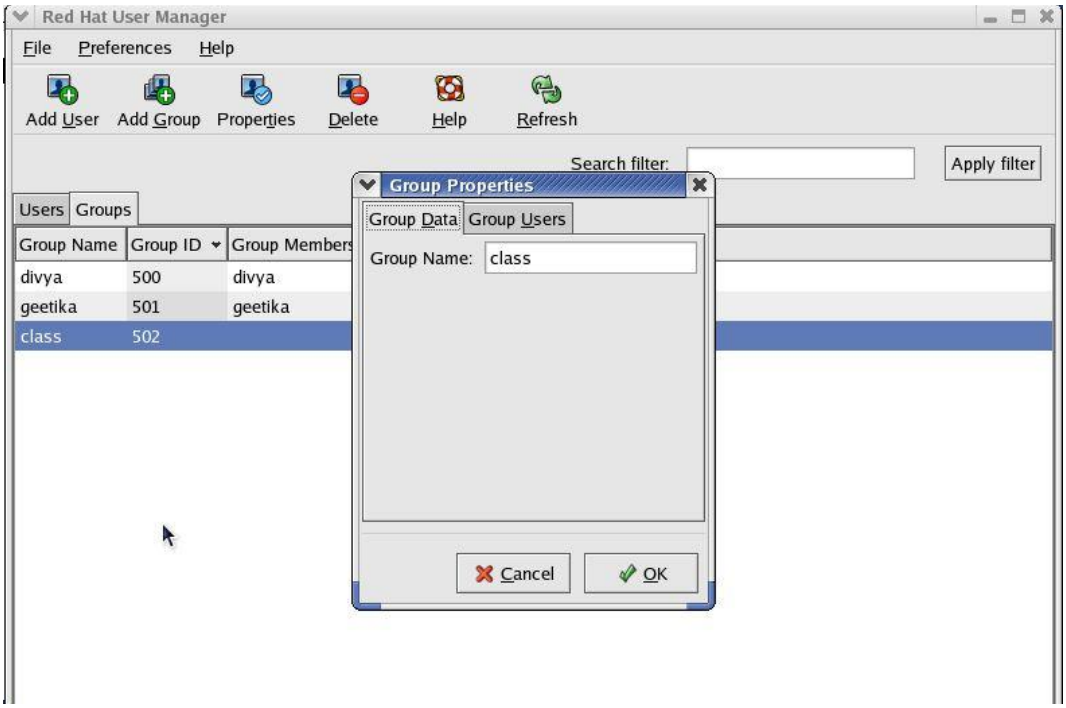
- Click the **Groups tab** in the User Manager window to work with groups.



- To create a group, click **Add Group** on the toolbar and specify the name of the group.

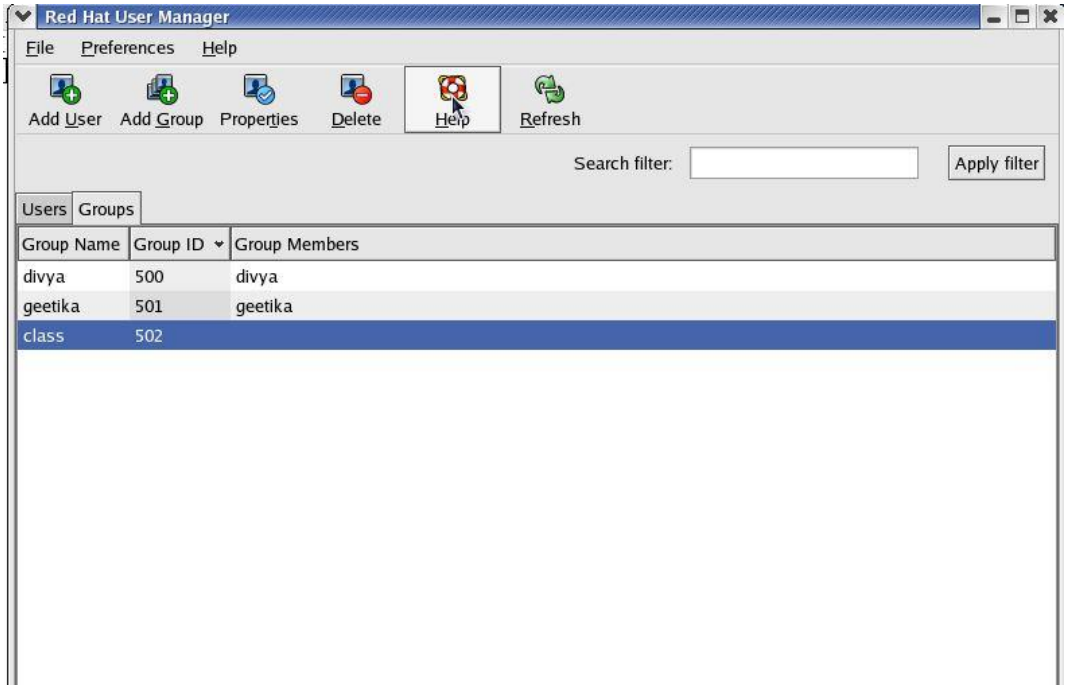



- To **change the name of a group or to add or remove users** from a group, highlight the group and click Properties on the toolbar.




## Help:

- The User Manager provides extensive help. To access it, click Help on the toolbar.
- When you are done working with users and groups, close the window.




useradd: Adds a User Account

The **useradd utility** adds a new user account to the system. By default, useradd assigns the next highest unused user ID to a new account and specifies bash as the user's login shell. This command creates the user's home directory (in /home), specifies the user's group ID, and puts the user's full name in the comment field: **# useradd -g 500 -c "Alex Watson" alex**

userdel: Removes a User Account

The **userdel utility** deletes user accounts. This command removes alex's account and his home directory hierarchy: **# userdel -r alex.** To turn off a user's account temporarily, you can use **usermod** to change the expiration date for the account. Because it specifies that his account expired in the past (December 31, 2009), the following command line prevents alex from logging in:

**# usermod -e "12/31/09" alex**

groupadd: Adds a Group

Just as useradd adds a new user to the system, **groupadd** adds a new group by adding an entry for it in **/etc/group.** creates a new group named rtfm: **# groupadd -g 1024 rtfm**

Unless you use the **–g** option to assign a group ID, the system picks the next available sequential number greater than 500. The **–o** option allows the group ID to be nonunique if you want to have multiple names for the same group ID.

## 10.16 Backing Up Files

The backup copies are vital in three instances:

1) When the system malfunctions and files are lost,
2) When a catastrophic disaster (fire, earthquake, and so on) occurs,
3) When a user or the system administrator deletes or corrupts a file by accident.

Even when you set up RAID, you still need to back up files. Although **RAID provides fault tolerance** (helpful in the event of disk failure), it does not help when a catastrophic disaster occurs

or when a file is corrupted or accidentally removed. You must back up filesystems on a regular basis. Backup files are usually kept on magnetic tape or some other removable media. A **full backup** makes copies of all files, regardless of when they were created or accessed. An **incremental backup** makes copies of those files that have been created or modified since the last (usually full) backup. The more people using the system, the more often you should back up the filesystems. One popular schedule is to perform an incremental backup one or two times a day and a full backup one or two times a week.

## Choosing a Backup Medium

If the local system is connected to a network, you can write your backups to a tape drive on another system. This technique is often used with networked computers to avoid the cost of having a tape drive on each computer in the network and to simplify management of backing up many computers in a network. Most likely you want to use a tape system for backups. Other options for holding backups are writable CDs, DVDs, and removable hard disks. These devices, although not as cost-effective or able to store as much information as tape systems, offer convenience and improved performance over using tapes.

## Backup Utilities

A number of utilities help you back up the system, and most work with any media. Most Linux backup utilities are based on one of the archive programs—**tar or cpio**— and augment these basic programs with bookkeeping support for managing backups conveniently. You can use any of the **tar, cpio, or dump/restore** utilities to construct full or partial backups of the system. Each utility constructs a large file that contains, or archives, other files. In addition to file contents, an archive includes header information for each file it holds.

The **amanda utility (Advanced Maryland Automatic Network Disk Archiver)**, one of the more popular backup systems, uses dump or tar and takes advantage of Samba to back up Windows systems. The amanda utility backs up a LAN of heterogeneous hosts to a single tape drive. You can use yum to install amanda.

### tar: Archives Files

The **tar (tape archive)** utility stores and retrieves files from an archive and can compress the archive to conserve space. If you do not specify an archive device, tar uses standard output and standard input. For displaying the options: **# tar --help | less**

It combines single-letter options into a single command-line argument: **# tar -ztvf /dev/st0**. This command uses descriptive words for the same options: **# tar --gzip --list --verbose --file /dev/st0**. Both commands tell tar to generate a (v, verbose) table of contents (t, list) from the tape on /dev/st0 (f, file), using gzip (z, gzip) to decompress the files.

| Option | Effect |
|---|---|
| **--append (-r)** | Appends files to an archive |
| **--catenate (-A)** | Adds one or more archives to the end of an existing archive |
| **--create (-c)** | Creates a new archive |
| **--delete** | Deletes files in an archive (not on tapes) |
| **--diff (-d)** | Compares files in an archive with disk files **--extract** |

| | |
|---|---|
| **--extract (–x)** | Extracts files from an archive |
| **--help** | Displays a help list of tar options |
| **--list (–t)** | Lists the files in an archive |
| **--update (–u)** | Like the –r option, but the file is not appended if a newer version is already in the archive |

### cpio: Archives Files

The **cpio (copy in/out)** program is similar to tar but can use archive files in a variety of formats, including the one used by tar. Normally cpio reads the names of the files to insert into the archive from standard input and produces the archive file as standard output. When extracting files from an archive, cpio reads the archive as standard input.

## Performing a Simple Backup

When you prepare to make a major change to a system, such as replacing a disk drive or updating the Linux kernel, it is a good idea to archive some or all of the files so you can restore any that become damaged if something goes wrong. For this type of backup, tar or cpio works well. For example, if you have a SCSI tape drive as device **/dev/st0** that is capable of holding all the files on a single tape, you can use the following commands to construct a backup tape of the entire system:

**# cd /**

**# tar –cf /dev/st0 .**

The tar command then creates an archive (**c)** on the device **/dev/st0 (f). # tar –cjf /dev/st0 .** You can back up the system with a combination of find and cpio. These commands create an output file and set the I/O block size to 5120 bytes (the default is 512 bytes):

**# cd /**

**# find . –depth | cpio –oB > /dev/st0**

This command restores the files in the /home directory from the preceding backup. The options extract files from an archive (–i) in verbose mode, keeping the modification times and creating directories as needed.

**# cd /**

**# cpio –ivmd /home/* < /dev/st0**

## dump, restore: Back Up and Restore Filesystems

The **dump utility**, which first appeared in UNIX version 6, backs up either an entire filesystem or only those files that have changed since the last dump. The **restore utility** restores an entire filesystem, an individual file, or a directory hierarchy. This command backs up all files (including directories and special files) on the root (/) partition to SCSI tape 0. Frequently there is a link to the active tape drive, named /dev/tape, which you can use in place of the actual entry in the /dev directory.

**# dump -0uf /dev/st0 /**

The option specifies that the entire filesystem is to be backed up (a full backup). There are ten dump levels: 0–9. Zero is the highest (most complete) level and always backs up the entire filesystem. Each additional level is incremental with respect to the level above it. For example, 1 is incremental to 0 and backs up only files that have changed since the last level 0 dump; 2 is incremental to 1 and backs up only files that have changed since the last level 1 dump; and so on. The **u option** updates

the **/etc/dumpdates** file with filesystem, date, and dump level information for use by the next incremental dump.

- The **f option** and its argument write the backup to the device named **/dev/st0.**
- The **u option** updates the **/etc/dumpdates** file with filesystem, date, and dump level information for use by the next incremental dump.
- The **f option** and its argument write the backup to the device named **/dev/st0.** This command makes a partial backup containing all files that have changed since the last level 0 dump. The first argument is a 1, specifying a level 1 dump:

  **# dump -1uf /dev/st0 /**
- To restore an entire filesystem from a tape, first restore the most recent complete (level 0) backup. Perform this operation carefully because restore can overwrite the existing filesystem. When you are logged in as Superuser, cd to the directory the filesystem is mounted on and give this command: **# restore -if /dev/st0**

  **i: Interactive mode**

  **f: specifies the name of the device that the backup medium is mounted on.**

## 10.17 Scheduling Tasks

It is a good practice to schedule certain routine tasks to run automatically. For example, you may want to remove old core files once a week, summarize accounting data daily, and rotate system log files monthly.

### crond and crontab: Schedule Routine Tasks

Using **crontab**, you can submit a list of commands in a format that can be read and executed by **crond**. Working as **Superuser**, you can put commands in one of the **/etc/cron.*** directories to be run at intervals specified by the directory name, such as cron.daily.

### at: Runs Occasional Tasks

Like the cron utility, at allows you to run a job sometime in the future. Unlike cron, at runs a job only once.

### System Reports

Many utilities report on one thing or another. The **who, finger, ls, ps**, and other utilities generate simple end-user reports. In some cases, these reports can help with system administration.

### vmstat: Reports Virtual Memory Statistics

The **vmstat utility** (procps package) generates virtual memory information along with (limited) disk and CPU activity data.

```
$ vmstat 3 7
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  2      0 684328  33924 219916    0    0   430   105 1052   134  2  4 86  8
 0  2      0 654632  34160 248840    0    0  4897  7683 1142   237  0  5  0 95
 0  3      0 623528  34224 279080    0    0  5056  8237 1094   178  0  4  0 95
 0  2      0 603176  34576 298936    0    0  3416   141 1161   255  0  4  0 96
 0  2      0 575912  34792 325616    0    0  4516  7267 1147   231  0  4  0 96
 1  2      0 549032  35164 351464    0    0  4429    77 1120   210  0  4  0 96
 0  2      0 523432  35448 376376    0    0  4173  6577 1135   234  0  4  0 95
```

vmstat column heads

This shows virtual memory statistics in 3-second intervals for seven iterations. The first line covers the time since the system was last booted; the rest of the lines cover the period since the previous line.

| | |
|---|---|
| **procs** | Process information |
| | r Number of waiting, runnable processes |
| | b Number of blocked processes (in uninterruptable sleep) |
| **Memory** | Memory Information in Kilobytes |
| | swpd Used virtual memory |
| | free Idle memory |
| | buff Memory used as buffers |
| | cache Memory used as cache |
| **swap** | System paging activity in kilobytes per second |
| | si Memory swapped in from disk |
| | so Memory swapped out to disk |
| **io** | System I/O activity in blocks per second |
| | bi Blocks received from a block device |

| | |
|---|---|
| | bo Blocks sent to a block device |
| **system** | Values are per second |
| | in Interrupts (including the clock) |
| | cs Context switches |
| **cpu** | Percentage of total CPU time spent in each of these states |
| | us User (nonkernel) |
| | sy System (kernel) |
| | id Idle |
| | wa Waiting for I/O |

top: Lists Processes Using the Most Resources

The **top utility** is a useful supplement to ps. At its simplest, top displays system information at the top and the most CPU-intensive processes below the system information. The top utility updates itself periodically; type **q to quit.**

```
$ top
top - 21:30:26 up 18 min,  2 users,  load average: 0.95, 0.30, 0.14
Tasks:  63 total,   4 running,  58 sleeping,   1 stopped,   0 zombie
Cpu(s): 30.9% us, 22.9% sy, 0.0% ni,  0.0% id, 45.2% wa, 1.0% hi, 0.0%si
Mem:   1036820k total,  1032276k used,     4544k free,    40908k buffers
Swap:  2048276k total,        0k used,  2048276k free,   846744k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1285 root      25   0  9272 6892 1312 R 29.3  0.7   0:00.88 bzip2
 1276 root      18   0  3048  860 1372 R  3.7  0.1   0:05.25 cp
    7 root      15   0     0    0    0 S  0.7  0.0   0:00.27 pdflush
    6 root      15   0     0    0    0 S  0.3  0.0   0:00.11 pdflush
    8 root      15   0     0    0    0 S  0.3  0.0   0:00.06 kswapd0
  300 root      15   0     0    0    0 S  0.3  0.0   0:00.24 kjournald
 1064 mgs2      16   0  8144 2276 6808 S  0.3  0.2   0:00.69 sshd
 1224 root      16   0  4964 1360 3944 S  0.3  0.1   0:00.03 bash
 1275 mgs2      16   0  2840  936 1784 R  0.3  0.1   0:00.15 top
 1284 root      15   0  2736  668 1416 S  0.3  0.1   0:00.01 tar
    1 root      16   0  2624  520 1312 S  0.0  0.1   0:06.51 init
```

top: interactive commands

| Command | Function |
|---|---|
| **F** | Specify a sort field. |
| **h or ?** | Displays a Help screen. |
| **k** | Prompts for a PID number and type of signal and sends the process that signal. Defaults to signal 15 (SIGTERM); specify 9 (SIGKILL) only when 15 does not work. |
| **M** | Sorts processes by memory usage. |
| **O** | Specify a sort field. |
| **P** | Sorts processes by CPU usage (default). |
| **q** | Quits. |
| **S** | Prompts for time between updates in seconds. Use 0 for continuous updates. |
| **SPACE** | Updates the display immediately. |
| **T** | Sorts tasks by time. |
| **W** | Writes a startup file named ~/.toprc so that next time you start top, it uses the same parameters it is currently using. |

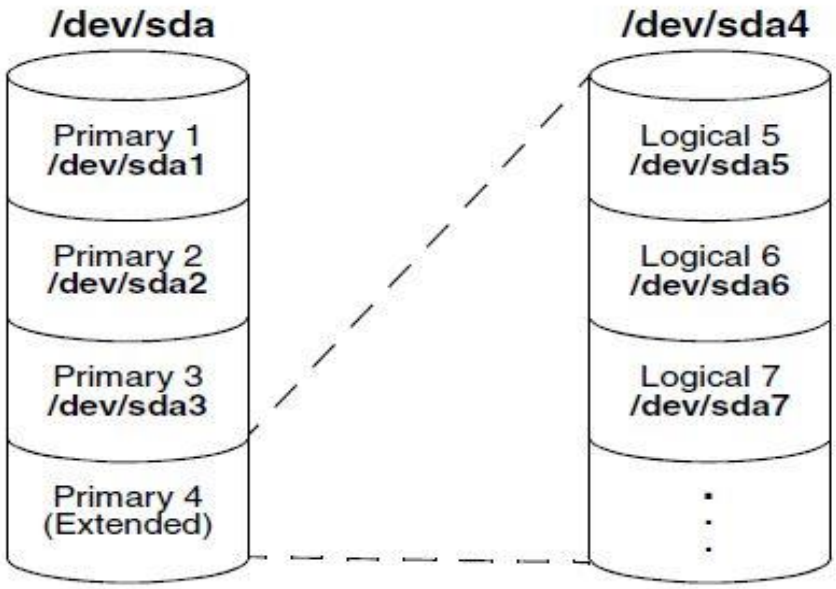
parted: Reports on and Partitions a Hard Disk

- The **parted (partition editor) utility** reports on and manipulates hard disk partitions.

The following example shows how to use parted from the command line

```
# parted /dev/sda print
Disk geometry for /dev/sda: 0kB - 165GB
Disk label type: msdos
Number  Start   End     Size    Type      File system  Flags
1       32kB    1045MB  1045MB  primary   ext3         boot
2       1045MB  12GB    10GB    primary   ext3
3       12GB    22GB    10GB    primary   ext3
4       22GB    165GB   143GB   extended
5       22GB    23GB    1045MB  logical   linux-swap
6       23GB    41GB    18GB    logical   ext3
7       41GB    82GB    41GB    logical   ext3
Information: Don't forget to update /etc/fstab, if necessary.
```

Partitions




parted: Reports on and Partitions a Hard Disk

The print command displays the following columns:

- **Number—**The minor device number of the device holding the partition. This number is the same as the last number in the device name. In the example, 5 corresponds to **/dev/sda5**.
- **Start—**The location on the disk where the partition starts. The parted utility specifies a location on the disk as the distance (in bytes) from the beginning of the disk. Thus partition 3 starts 12 gigabytes from the beginning of the disk.
- **End—**The location on the disk where the partition stops. Although partition 2 ends 12 gigabytes from the beginning of the disk and partition 3 starts at the same location, parted takes care that the partitions do not overlap at this single byte.
- **Size—**The size of the partition in kilobytes (kB), megabytes (MB), or gigabytes (GB).
- **Type—**The partition type: primary, extended, or logical.
- **File system—**The filesystem type: ext2, ext3, fat32, linux-swap, and so on.
- **Flags—**The flags that are turned on for the partition, including boot, raid, and lvm. In the example, partition 1 is bootable

## Summary:

- Superuser can use certain tools, such as sudo, to give specific users permission to perform tasks that are normally reserved for Superuser.
- To bring a system up in rescue mode, boot the system from the first installation CD, the Net Boot CD, or the install DVD.
- SELinux can be in one of three states: enforcing, permissive and disabled.
- The system-config-selinux utility displays the SELinux Administration window, which controls SELinux.
- By default, Fedora systems boot to graphical multiuser mode (runlevel 5).
- The system-config-services utility displays the Service Configuration window.
- The shutdown and halt utilities perform the tasks needed to bring the system down safely.
- The key combination CONTROL-ALT-DEL Reboots the System

- Most of the Fedora/RHEL configuration tools are named system-config-*.
- When you set up a server, you frequently need to specify which clients are allowed to connect to the server. Sometimes it is convenient to specify a range of IP addresses, called a subnet.
- RHEL uses the xinetd daemon, a more secure replacement for the inetd superserver that was originally shipped with 4.3BSD.
- You may secure a server either by using TCP wrappers or by setting up a chroot jail.
- An ordinary file stores user data, such as textual information, programs, or images, such as a jpeg or tiff file.
- A character device is any device that is not a block device. Examples of character devices include printers, terminals, tape drives, and modems.

## Keywords

- **System Administrator**: A system administrator should be available to help users with all types of system-related problems—from logging in to obtaining and installing software updates to tracking down and fixing obscure network issues.
- **su**: The su (substitute user) utility can create a shell or execute a program with the identity and permissions of a specified user.
- **Consolehelper:** The consolehelper utility can make it easier for someone who is logged in on the system console but not logged in as root to run system programs that normally can be run only by root.
- **Trojan House:** A Trojan horse is a program that does something destructive or disruptive to a system while appearing to be benign.
- **runlevel utiity**: The runlevel utility displays the previous and current runlevels. This utility is a transitional tool; it provides compatibility with SysVinit.
- **Booting:** Booting a system is the process of reading the Linux kernel into system memory and starting it running.
- **rpcinfo:** The rpcinfo utility displays information about programs registered with rpcbind and makes RPC calls to programs to see if they are alive.
- **Inode:** An inode is a data structure, stored on disk, that defines a file's existence and is identified by an inode number.
- **Sockets:** Sockets allow unrelated processes on the same or different computers to exchange information.
- **Hotplug:** The hotplug system allows you to plug a device into the system and use it immediately.
- **Mount point:** The mount point for the filesystem/directory hierarchy that you are mounting is a directory in the local filesystem.
- **umount:** The **umount** utility unmounts a filesystem as long as it does not contain any files or directories that are in use (open).

## Self Assessment

1. su stands for

A. substitute user

B. switch user

C. substandard user

D. None of the above

2. Which of these tools gives you another user's privileges?

A. kill

B. su

C. consolehelper

D. None of the above

3. Which of these tools runs programs as a root?

A. kill

B. su

C. consolehelper

D. None of the above

4. Who is a superuser in Linux environment?

A. Root

B. Normal user

C. Machine

D. None of the above

5. What are the modes of SELinux?

A. Enforcing

B. Permissive

C. Disabled

D. All of the above

6. The policies of SELinux are

A. Targeted

B. MLS

C. Strict

D. All of the above

7. Which of these key combinations reboots the system?

A. CTRL-ALT-HOME

B. CTRL-DEL-END

C. CTRL-ALT-DEL

D. CTRL-TAB-DEL

8. Which of these utility displays the kernel ring buffer?

A. chsh

B. clear

C. dmesg

D. None of the above

9. Which of these utility creates a new filesystem on device?

A. mkfs

B. chsh

C. dmesg

D. clear

10. Which superserver listens for network connection?

A. xinted

B. Machine

C. Fedora

D. None of the above

11. How can we secure a server?

A. By using TCP wrappers

B. By setting up a chroot jail

C. By using both of the above

D. None of the above

12. The user's interactive non-login shell initialization script is located in _____ file.

A. ~/.bash_profile

B. ~/.bashrc

C. /dev

D. None of the above

13. Which of these is known as a bit bucket?

A. /dev/empty

B. /dev/bucket

C. /dev/null

D. None of the above

14. The _____ option causes Linux not to mount the filesystem automatically.

A. noauto

B. nosuid

C. nomount

D. None of the above

15. The _____ backup makes copies of all the files.

A. Full

B. Incremental

C. Decremental

D. Half

## Answer for Self Assessment

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | A | 2. | B | 3. | C | 4. | A | 5. | D |
| 6. | D | 7. | C | 8. | C | 9. | A | 10. | A |
| 11. | C | 12. | B | 13. | C | 14. | A | 15. | A |

## Review Questions

1. What is a well-maintained system? What kind of powers a superuser has?
2. How can we gain/grant the superuser privileges?
3. What are system administration tools? Explain.
4. What is a rescue mode? How can we avoid trojan horse?
5. What is security enhanced linux? What are states and policies of SELinux?
6. What is a run-level? Explain various utilities associated with this.
7. What is Fedora/RHEL configuration tools? Explain.
8. What are command line utilities?
9. What are standard rules in configuration files? How can we specify cilents?
10. How can we secure a server?
11. What are important files and directories?
12. What kind of files are supported by Linux?
13. What is fstab? How can we keep track of filesystems? Explain the columns of fstab file.
14. What is a special file in Linux?
15. Explain various types of file systems in Linux?
16. What is backing up of files? How can we choose a backup medium? Explain various backup utilities.
17. How can we schedule tasks? What are vmstat column heads?

## Further Readings

Mark G. Sobell, A Practical Guide to Fedora and RedHat Enterprise Linux, Fifth Edition, Prentice Hall

### Web Links

https://www.beyondtrust.com/resources/glossary/superuser-superuser-accounts#:~:text=In%20Linux%20and%20Unix%2Dlike,any%20permissions%20for%20other%20users.

https://searchsecurity.techtarget.com/definition/sudo-superuser-do