Linux

History of Linux

Linux is a free operating system. It is a developed by **'Linus Torvalds'**. He uploaded his improved version of the **'MINIX' Operating System** onto the Internet in **1991,** a small but self-contained kernel for the 80386 processor and the first true 32-bit processor in Intel's range of PC-compatible CPUs. Linux operating system is the product of the Internet. The source code should always be distributed along with the software. Linux is a UNIX work-alike.

Linux is a multi-user, multi-tasking operating system that runs on many platforms including INTEL. It implements a superset of the **POSIX standard**. Linux is also capable of interaction with other operating system, including computing operating system from APPLE, MICROSOFT and NOVELL

The GNU Project

Linux has also gained from its association with the GNU Project. The **GNU Project** started in **1984** to develop a complete **free UNIX-like operating system**. Variants of the GNU System using Linux as the Kernel (Core of the Operating System) are now widely used though often called "Linux", they are more accurately called "GNU Linux System".

The founder of GNU, '**Richard Stallman**', is seen by many to be one of the principal architects in this new way of thinking about software development.

The GNU Project is founder by the free software foundation whose description follows:
The Free Software Foundation (FSF) is dedicated to eliminating restrictions on copying, redistribution, understanding, and modification of computer programs. We do this by promoting the development and use of free software in all areas of computing- but most particularly helping to develop the GNU Operating System.

Principal among GNU's many contributions to computer science are:

1. **The GNU Emacs Editor:**
   One of the most powerful editors you can find today.
2. **The GNU 'C' Compiler:**
   A Compiler that now support 7 language and over 30 different architectures.
3. **Ghost Script:**
   An interpreter for the PostScript and PDF graphics language.
4. **The Gimp:**
   The GNU Image Manipulation Program a Photoshop like graphics application only better.
5. **Gnome:**
   The GNU desktop a graphical user interface for the desktop that includes many different Useful applications.

Linux Distributions Or Flavours

---

1. Red Hat Linux
2. Caldera Open Linux
3. SuSE Linux
4. Turbo Linux
5. Debian Linux
6. Slackware Linux
7. Mandrake Linux

Features of Linux

---

1. **Free:**
   Linux is available for free on the Internet. We don't have to pay anything to download it. A lot of distributions of Linux are available for free download.
2. **Open Source:**
   Open Source means we get not only the executables but also the source code. The user can access the source code and make improvements with it.
3. **Well Documented:**
   The 'HOW-Tos', as they are called, are extensive and very simple.
4. **Customisable:**
   Linux users have the option of opting for using other software other than those bundled with the installation kit. These other software may either be commercial software or free software available off the net.
5. **Hardware:**
   Linux can run on almost any hardware, be it 386, 486, Pentium MMX, Pentium II, Sparc, Dec Alpha or even Motorola 68000 Series.
6. **Multi-Tasking:**
   Linux is a multi-tasking system. A single user can run multiple programs at the same time. Each task is called a process. This means that a user can give the system a command to be run in the background while doing more important work in the foreground.
7. **Multi-User:**
   This means that more than one user can use the system at the same time. The multi user concept stems directly from the multi-tasking aspect.
8. **Multiple Virtual Consoles:**
   We can have Multiple Virtual Terminals on the server. This can be done by pressing a key Combination.

   In Linux it is

CTRL + ALT + any of F1 to F6 –	for character mode terminals
CTRL + ALT + F7 -	or GUI terminals

9. **TCP/IP Networking:**
   TCP/IP networking support is build into the 'Kernel' itself. Linux is among the best operating system in terms of networking. It includes programs like 'Telnet', 'FTP', 'rlogin', 'rsh' and other such programs.
10. **High Level Security:**
    One of the main advantages of Linux is that it provides a very high level of security by using user authentication. It also stores passwords in an encrypted form. The password once encrypted cannot be decrypted.
11. **Programming Supports**:
    Linux Provides Programming Support for FORTRAN, C, C++, TCL/TK, Perl and many other languages.
12. **GUI:**
    Linux has a very good Graphical user Interface (GUI) with window manager like KDE. GNOME has taken GUI for Linux one step a head and given it a very gook look.
13. **Stable Operating System:**
    Linux is a complete operating system that is stable. This means that the malfunctioning of an application is not likely to being the system down.
14. **Reliable:**

Linux Server is very reliable. The regular reboots seen in the case of the other Operating System.

15. **Web Server:**
    Linux can be used to run a web server such as 'Apache' to serve application protocols such as HTTP or FTP.
16. **Multi-Processor Support:**
    Linux supports multiple Processors.

Hardware Requirement of Linux

Linux runs on a variety of computing platforms unmatched by any other operating system. A short list of major platforms includes:

- Intel x86
- PowerPC
- MIPS
- Sparc
- Motorola 680x
- Strong Arm
- DEC Alpha

The versions of Linux that run on these platforms vary in stability and completeness with the Intel x86 versions as the most stable. The PowerPC and Alpha ports follow close behind in stability and compatibility. For instance, the PowerPC team will have support for the newest Macintosh models by the time you read this. That support includes advanced features such as USB, Firmware and support for the G3 Processors.

Minimal Linux machine can look like the following.

- 386 CPU
- 4 MB RAM
- Floppy Drive
- VGA Video

A Typical Linux workstation machine looks more like the followings:

- Pentium Class or Equivalent CPU
  (Such as an AMD K5 or Cyrix Cx586)
- 64 MB RAM
- Floppy Drive
- CD-ROM Drive
- 4 GB Hard Drive
- IDE or SCSI Disk (Small Computer System Interface)
- SVGA Graphics Card
- Ethernet Card or Modem
- Color Monitor

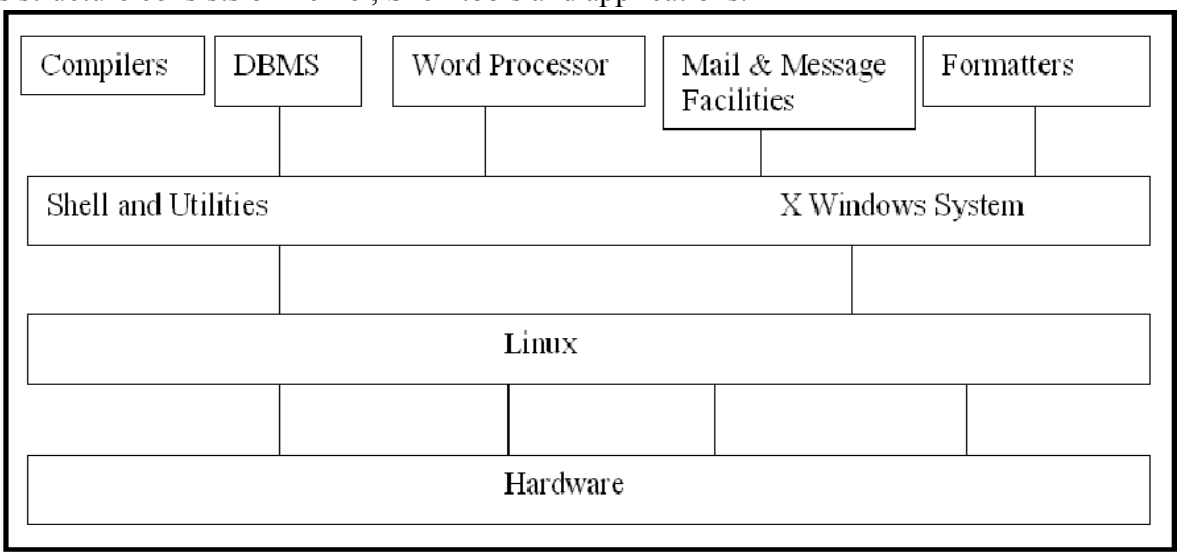
A Typical Linux server configuration looks something like the following:

- Pentium II Class Processor or Equivalent CPU
  (Such as AMD K6-2 or Cyrix M2)
- 128 MB RAM
- Floppy Drive

- CD-ROM Drive
- 9 GB SCSI hard disk or Multi drive RAID Array
- Ethernet Card or Modem
- VGA Graphics Card
- Tape Backup Drive

Structure of Linux System

Its structure consists of Kernel, Shell tools and applications.



The Linux operating system includes features from both BSD and UNIX system. The core of the operating system. Kernel implemented independent of both the UNIX system.

Linux supports many peripherals, which work real fast. Linux is not for Intel based platforms but has been ported to machines like PowerPC – including Apples, DEC alpha based machines, and MIPS based machines and 68k based machines. Linux supports programs called emulator that allow user to run code intended for other operating system.

The Parts of the LINUX System

---

1. **Kernel:**

   Kernel forms the core of the LINUX Operating System. This interacts with the hardware. It is loaded into the memory when a System is booted. Its functions are.

   - Managing the System Resources.
   - Allocating Time for different Users and Processes.
   - Deciding Process Priorities and performing them.

   Kernel does not interact with the user directly instead, it starts up a separate file called "Shell".

2. **Shell:**

   Linux's shell interacts with the user and the computer. Some of its features are as follows.

   I. **Interactive Processing:**

   Communication between the user and the system takes the form of interactive dialogue with the shell.

   II. **Background Processing:**

   There are processes, which are time consuming and which are non-interactive in nature. Such processes can be done in the background while we can continue using the system to do other tasks.

III. **Input/Output Redirection:**
Programs can be instructed to take input from a file other then the standard input i.e. the keyboard and send the output to a file instead of the standard output i.e. the visual display unit. This is called redirection. Simultaneous use of input and output redirection is possible.

IV. **Pipes:**
Simple programs can be developed to do complex operations with minimum effort using pipes.

V. **Wild Card Patterns:**
File matching a particular pattern can be grouped together and actions may be performed on it.

VI. **Shell Scripts:**
The Shell Scripts contain a sequence of commands to be performed with a single filename. The script can be executed using that filename.

VII. **Shell Variables**:
By storing data in variables we can control the behaviors of shell as well as that of programs and utilities.

VIII. **Programming Language Constructs:**
Shell has features, which enabled. It to be used as a programming language. These help in building complex shells to perform complex.

Note: There are different types of shell available. They are Bourne Shell, C Shell, Korne Shell and Restricted Shell. The Shell that is officially distributed with the LINUX System and widely is the bash Shell.

IX. **Tools And Applications:**
Linux supports business application oriented packages like word processor, Electronic spreadsheets and databases etc.

The .profile File

Linux allows us to create customize environment. A few of the environmental variables like HOME, LOGNAME are set as soon as we login in the **.profile** file. The other variables can be set by the user. The .profile file is similar to the autoexec.bat file in DOS. This file could be executed only if exists in the HOME directory.

The **'/ect/stdprofile'** file is copies by the system administrator into user's home directory as the .profile file when we enter newly into the multi-user system. This file will contain the standard path and the type of the terminal to be used. We can change the settings by changing the environmental variables suitably.

The '.bash_profile' Shell Script

The shell environment in Linux can be customized. In the 'bash' shell, this can be done by using the **'.bash_profile'** shell script.

The contents of the **'.bash_profile'** are read and executed every the user logs in. Every user has a copy of the '.bash_profile' file of the user's home directory. This copy of the **'.bash_profile'** is used to customize the user's environment.

The **'.bash_profile'** shell script is the first shell script to be executed when a user logs in. It is the personal copy of the '.bash profile' shell script that is executed. This file can be customized to the user's preferences. Programs that are to be executed when the user logs in are put here. The environment variables can also be altered to the user's preferences.

Process

A process is a program in execution. A process is more than the program code, which is sometimes, knows as the **text section**. It also includes the current activity, as represented by the value of the **program counter** and the contents of the processor's registers. In addition, a process general includes the process **stack**, which contains temporary data (such as method parameters, return addresses, and local variables), and a **data section**, which contains global variables.

We emphasize that a program by itself is not a process; a program is a passive entity, such as the contents of a file stored on disk, whereas a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources.

Process State

The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

- **New:** The process is being created.
- **Running:** Instructions are being executed.
- **Waiting:** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- **Ready:** The process is waiting to be assigned to a processor.
- **Terminated:** The process has finished execution.

Only one process can be running on any processor at any instant, although many processes may be ready and waiting.

Process Control Block (PCB)

Each process is represented in the operating system by a **Process Control Block (PCB)** – also called a **Task Control Block**. It contains many pieces of information associated with a specific process, including these:

- **Process State:** The state may be new, ready, running, waiting, halted and so on
- **Program Counter:** The counter indicates the address of the next instruction to be executed for this process.
- **CPU Registers:** The registers vary in number and type, depending on the computers architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.
- **CPU-Scheduling Information:** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-Management Information:** This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used are the operating system.
- **Accounting Information:** This information includes the amount of CPU and real time used, time limits, account numbers, job or process number and so on.
- **I/O Status Information:** The information includes the list of I/O devices allocated to this process, a list of open files, and so on.

Process Creation

The creating process is called a parent process, whereas the new processes are called the children of that process. Each of these new processes may in turn create other processes, forming a tree of processes.

A process will need certain resources (such as CPU time, memory, files, I/O devices) to accomplish its task. When a process creates a sub process that sub process may be able to obtain its resource directly from the operating system, or it may be constrained to a subset of the resources of the parent process. The parent may have to partition its resources among its children, or it may be able to share some resource (such as memory of files) among several of its children. Restricting a child process to a subset of the parent's resources prevents any process from overloading the system by creating too many sub processes.

When a process creates a new process, two possibilities exist in terms of execution:

1. The parent continues to execute concurrently with its children.
2. The parent waits until some or all of its children have terminated.

There are also two possibilities in terms of the address space of the new process:

1. The child process is a duplicate of the parent process.
2. The child process has a program loaded into it.

These different implementations let us consider the Linux operating system. In Linux, each process is identified by **process identifier**, which is a unique integer. A new process is created by the **fork** system call. The new process consists of a copy of the address space of the original process. This mechanism allows the parent process to communicate easily with its child process. Both processes (the parent and the child) continue execution at the instruction after the **fork** system call, with one difference: The return code for the **fork** system call is zero for the new (child) process, whereas the (nonzero) process identifier of the child is returned to the parent.

The **execlp** system call is used after a fork system call by one of the two processes to replace the process memory space with a new program. The **execlp** system call loads a binary file into memory-destroying the memory image of the program containing the **execlp** system call- and states its execution.

File Systems

Linux retains UNIX's standard file-system model.

1. The Virtual File System

The Linux VFS is designed around object-oriented principles. It has two components: a set of definition that define what a file object is allowed to look like, and a layer of software to manipulate those objects. The three main object types defined by the VFS are the inode-object and the file-object structures, which represent individual files, and the file-system-object, which represents an entire file system.

For each of these three types of object, the VFS defines a set of operations that must be implemented by that structure. Every object of one of these types contains a pointer to a function table. The function table lists the addresses of the actual functions that implement those operations for that particular object.

2. The Linux ext2fs File System

The standard on-disk file system used by Linux is called ext2fs, for historical reasons. Linux was originally programmed with a Minix-Compatible file System, to ease exchanging data with the Minix development system, but that file system was severely restricted by 14-character file-name limits and the maximum file-system size of 64 MB. The Minix file system was superseded by a new file system, which was christened the **extended file system (extfs)**. A later redesign of this file system to improve performance and scalability and to add a few missing features led to the **second extended file system (ext2fs)**

Ext2fs has mush in common with the BSD Fast File System (ffs). The main differences between ext2fs and ffs lie in the disk-allocation policies. In ffs, the disk is allocated to files in blocks of 8 KB, with blocks being subdivided into fragments of 1 KB to store small files or partially filled blocks at the end of a file. In contrast, ext2fs does not use fragments at all, but performs all its allocations in smaller units. The default block size on ext2fs in 1 KB, although 2 KB and 4 KB blocks are also supported.

3. **The Linux Proc File System**

The Linux VFS is sufficiently flexible that it is possible to implement a file system that does not store data persistently at all, but rather simply provides an interface to some other functionality. The Linux **process file system**, known as the **proc file system**.

A proc file system is not unique to Linux. SVR4 UNIX introduced a proc file system as an efficient interface to the kernel's process debugging support: Each subdirectory of the file system corresponded not to a directory on any disk, but rather to an active process on the current system. A listing of the file system reveals one directory per process, with the directory name being the ASCII decimal representation of the process unique process identifier (PID).

Linux implements such a proc file system, but extends it greatly by adding a number of extra directories and text files under the file systems root directory. These new entries correspond to various statistics about the kernel and grams to access this information as plain text files, which the standard UNIX user environment provides powerful tools to process.

File

A LINUX file is storehouse of information –it is simply a sequence of characters. A file contains exactly those bytes that you put into it, whether it represents a source program, other text or executable code.

File Type

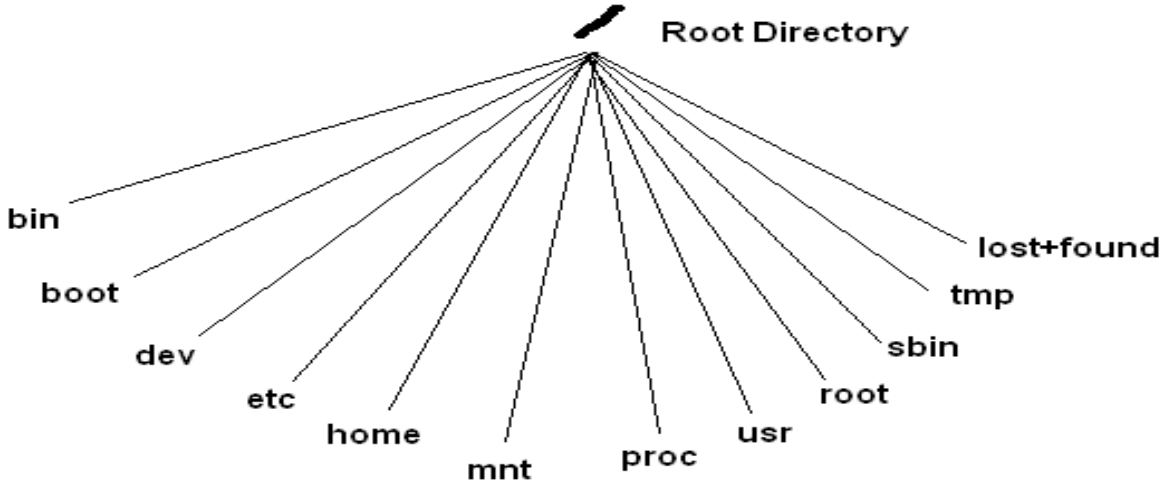
1. **Ordinary File**

   This is the traditional definition of a file. It consists of a stream of data resident on some permanent magnetic media. This includes all data, source programs, objects and executable code, all LINUX programs, as well as any files created by the user. This file is also referred to as a **regular file**.
2. **Directory File**

   A directory can be defined as a storehouse where other programs and sub-directories reside. A directory contains no data, but keeps an account of all the files and sub-directories that it contains. A directory file contains two fields – the name of a file and a pointer to a separate disk area, which contains the file attributes.
3. **Device File**

   The definition of a file has been broadened by LINUX to consider even physical devices as files. This definition includes printers, tapes, floppy, drives, hard disks and terminals.

The Structure of the File System



/bin - All the executable files
/boot - The 'Linux Kernel' or booting file.
/dev - the entire device files
/etc - The administrative files.
/home - All the user areas.
/lib - All the C sub routine files and parts of the C compiler.
/mnt - Mount Point for the floppy, CD drive or other hard disk partitions.
/proc - All the process information files.
/root - The root's user home area.
/sbin - All files, which are needed for configuring and running the system.
/tmp - All temporary files.
/usr - The user file system.
/var - All logs, spools for printing etc.

Internal Structure of File System

In the LINUX system, every file has a table associated with it, which is stored in a special area of the disk. This is called the **Identification node**, and is usually abbreviated to **i-node**. The i-node describes a file uniquely, and the structure of this table will be taken up shortly. A single disk may have more than one file system in it. A single file system can't span multiple disks.

Every file system consists of a sequence of blocks, each block consisting of 512 bytes. Some of these blocks are not allotted to the user, and are reserved exclusively for the use of the kernel. The file system breaks the disk into four segments.

a) The boot block

The first block is called the book block, which is normally unused by the file system, and set side for the booting procedure. The is true only for the main file system. For the other file systems, this block is left unused.

b) The super block

The second block is called the super block, is used to control the allocation of disk blocks. If you have been using a LINUX for some time, you would have noticed that, unlike MSDOS systems,

the machine is powered down by going through a format routine. The super block is that section of the disk, which needs to be updated in a periodic manner.

The super block contains details of the active file system. This block contains the size and status of the file system, the details of the free blocks and i-nodes. This includes a number, as well as a list of free blocks of the system, which can be used immediately. The super block doesn't contain a complete list of all the free blocks and i-nodes, but only as much as is required to meet current needs.

c) The i-node blocks

The i-node block includes, up to a number determined during the creation of the file system, and contains most information pertaining to files. Every file in the file system will invariably have an entry in this area, identified by a 64-bytes structure referred to as the i-note. The complete list of i-nodes is known, as the i-list. Every i-note is identified by unique number, which simply references the position of the i-node in the list. This number is called the i-number, and can be considered to be the internal name of a file.

An i-node is 64 bytes long. So in one physical block there are 8 i-nodes. Each i-node contains the following attributes of a file:

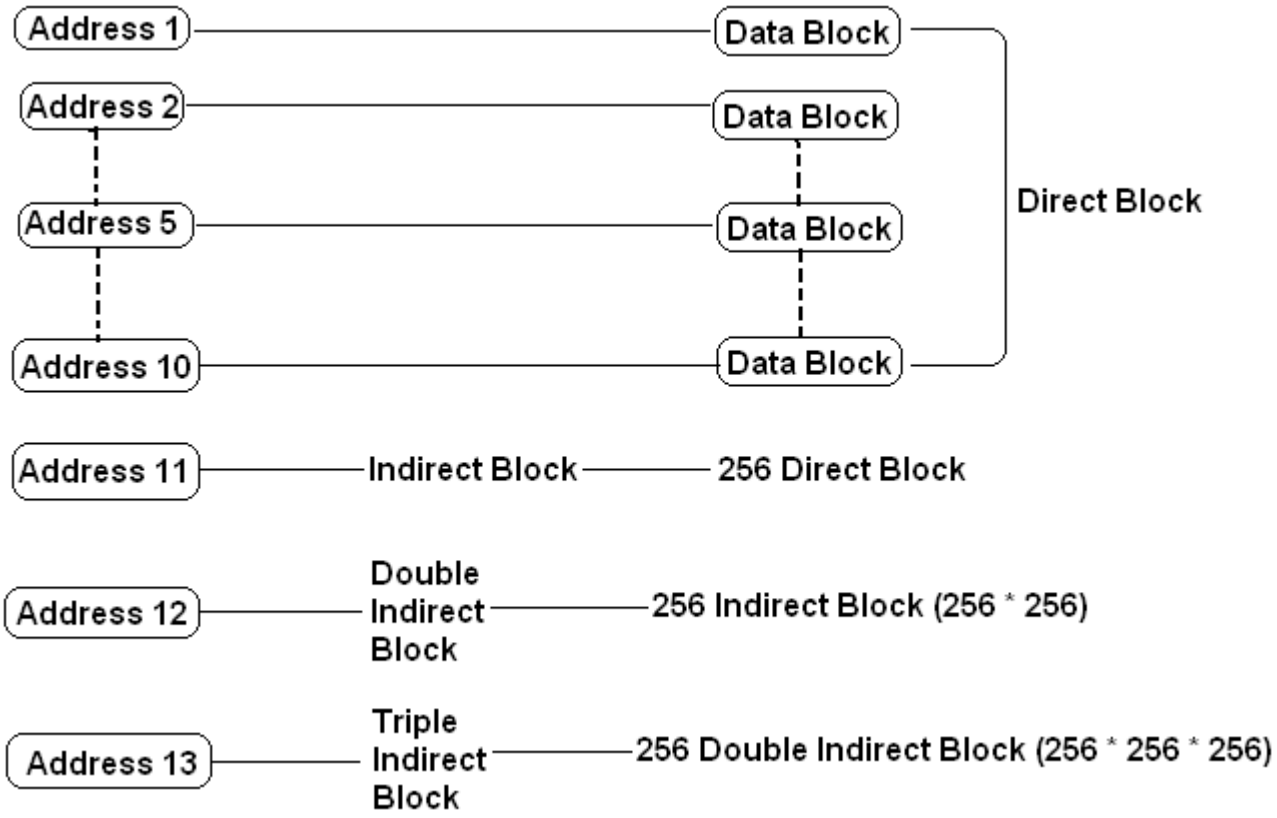
- The file type (regular, directory, etc.)
- The number of links (the number of aliases the file has)
- The owner (the user-id number of the owner)
- The group (the group-id number)
- The file mode (the triad of the three permissions)
- The number of bytes in the file
- The date and time of creation
- The date and time of last modification
- The date and time of the last access
- An array of 13 pointers to the file

Note: The name of the file is not stored in the i-node.

d) The data blocks

The data block is contains a long chain of blocks for storing the contents of files. The LINUX file system stores data in physical blocks of 512 bytes. These data blocks commence from the point the i-node blocks terminate. A LINUX file is a sequentially organized set of blocks scattered throughout the disk. It is the array of the 13 disk block addresses, which keeps track of all disk block containing the file segments.

The first ten pointers are direct addresses of a file blocks, they contain the addresses of the first ten storage blocks of a file. As a file grows beyond ten blocks, an 11^{th} data block is allocated to specify a disk block, which contains the addresses of the next 256 block of the file. This block is called the indirect block. When the file grows beyond this size, the 12^{th} block, known as the double indirect block is used. This block contains the address of another block, which in turn contains the addresses of 256 indirect blocks. Finally, if that space is not enough, the 13^{th} pointer, known as the triple indirect block is used. This block contains the address of another block, which in turn contains the address of 256 double indirect blocks, enhances the maximum possible file size to 16843018 (10 + 256 + 256*256 + 256*256*256) blocks. With one block taken as 1024 bytes, this works out around 17 GB.


Address 1
Data Block
Address 2
Data Block
Address 5
Data Block
Address 10
Data Block
Direct Block
Address 11
Indirect Block
256 Direct Block
Address 12
Double Indirect Block
256 Indirect Block (256 * 256)
Address 13
Triple Indirect Block
256 Double Indirect Block (256 * 256 * 256)

Linux Commands

---

1. LS

ls - list directory contents

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuSUX nor --sort. Mandatory arguments to long options are mandatory for short options too.

--color[=WHEN]	:	control whether color is used to distinguish file types. WHEN may be `never', `always', or `auto'
-d, --directory	:	list directory entries instead of contents
-F, --classify	:	append indicator (one of */@) to entries '*' for executables file '/' for directories '@' for symbolic links file
-l	:	use a long listing format
-L, --dereference	:	when showing file information for a symbolic link, show information for the file the link references rather than for the link itself
-r, --reverse	:	reverse order while sorting
-R, --recursive	:	list subdirectories recursively
-s, --size	:	print size of each file, in blocks
-S	:	sort by file size
-t	:	sort by modification time
-v	:	sort by version
-x	:	list entries by lines instead of by columns
-X	:	sort alphabetically by entry extension
-1	:	list one file per line
--help	:	display this help and exit
--version	:	output version information and exit

By default, color is not used to distinguish types of files. That is equivalent to using --color=none. Using the --color option without the optional WHEN argument is equivalent to using --color=always. With --color=auto, color codes are output only if standard output is connected to a terminal (tty).

List is show long format is…

d | rw-rw-r— | 2 | student | student | 34 | may 12 3:43 | my/

1 2 3 4 5 6 7 8

Explains the following groups in show the long format is

1. **File Type:** d' for directories, 'l' for symbolic links, '-' for regular files
2. **File Permission:** The file permission is display in three groups. Each group is contain three character (rwx – r for read , w for write and x for execute).
   First Three characters for **Owner Permission**
   Second Three characters for **Group Permissioin** and
   Third Three characters for **Other User.**
3. **Number of Links**

4. **File Owner**
5. **File Owner Group**
6. **File Size**
7. **Date and Time**
8. **File or Directory Name**

2. CAL

Cal - displays a calendar

cal [-my13] [[month] year]

DESCRIPTION

Cal displays a simple calendar. If arguments are not specified, the current month is displayed. The options are as follows:

-1 : Display single month output (use if cal was built with -3 as default to get older traditional output)
-3 : Display prev/current/next month output (use if cal was built with traditional -1 as default to get Newer improved output)
-m : Display Monday as the first day of the week. (The default is Sunday.)
-y : Display a calendar for the current year.

A single parameter specifies the year (1 - 9999) to be displayed; note the year must be fully specified: “cal 89 " will not display a calendar for 1989. Two parameters denote the month (1 - 12) and year. If no parameters are specified, the current month's calendar is displayed.

A year starts on Jan 1.

3. CAT

cat - concatenate files and print on the standard output

cat [OPTION] [FILE]...

DESCRIPTION

Concatenate FILE(s), or standard input, to standard output.

-b, --number-nonblank : number nonblank output lines
-E, --show-ends : display $ at end of each line
-n, --number : number all output lines
-s, --squeeze-blank : never more than one single blank line
-T, --show-tabs : display TAB characters as ^I
-v, --show-nonprinting : use ^ and M- notation, except for LFD and TAB
--help : display this help

With no FILE, or when FILE is -, read standard input.

4. CLEAR

clear - clear the terminal screen

clear

DESCRIPTION

clear clears your screen if this is possible. It looks in the environment for the terminal type and then in the term info database to figure out how to clear the screen.

5. MKDIR

mkdir - make directories

mkdir [OPTION] DIRECTORY...

DESCRIPTION

Create the DIRECTORY (ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

-m, --mode=MODE	:	set permission mode (as in chmod), not rwxrwxrwx - umask
-p, --parents	:	no error if existing, make parent directories as needed
-v, --verbose	:	print a message for each created directory
--help	:	display this help and exit
--version	:	output version information and exit

6. CD

cd - Change working directory

cd ?dirName?

DESCRIPTION

Change the current working directory to dirName, or to the home directory (as specified in the HOME environment variable) if dirName is not given. Returns an empty string.

7. CP

cp - copy files and directories

cp [OPTION]... SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.

-f, --force	:	if an existing destination file cannot be opened, remove it and try again
-i, --interactive	:	prompt before overwrite
-l, --link	:	link files instead of copying
--parents	:	append source path to DIRECTORY
-r	:	copy recursively, non-directories as files WARNING: use -R instead when you might copy special files like FIFOs or /dev/zero
--remove-destination	:	remove each existing destination file before attempting to open it (contrast with --force)
-R, --recursive	:	copy directories recursively
-s, --symbolic-link	:	make symbolic links instead of copying

-u, --update	:	copy only when the SOURCE file is newer than the destination file or when the destination file is missing
-v, --verbose	:	explain what is being done
--help	:	display this help and exit
--version	:	output version information and exit

8. LN

ln - make links between files

ln [OPTION]... TARGET [LINK_NAME]
ln [OPTION]... TARGET... DIRECTORY
ln [OPTION]... --target-directory=DIRECTORY TARGET...

DESCRIPTION

Create a link to the specified TARGET with optional LINK_NAME. If LINK_NAME is omitted, a link with the same basename as the TARGET is created in the current directory. Create hard links by default, symbolic links with --symbolic. When creating hard links, each TARGET must exist.

Mandatory arguments to long options are mandatory for short options too.

-d, -F, --directory	:	hard link directories (super-user only)
-f, --force	:	remove existing destination files
-i, --interactive	:	prompt whether to remove destinations
-s, --symbolic	:	make symbolic links instead of hard links
-v, --verbose	:	print name of each file before linking
--help	:	display this help and exit
--version	:	output version information and exit

9. MV

mv - move (rename) files

mv [OPTION]... SOURCE DEST

DESCRIPTION

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.

-f, --force	:	do not prompt before overwriting equivalent to --reply=yes
-i, --interactive	:	prompt before overwrite equivalent to --reply=query
--reply={yes,no,query}	:	specify how to handle the prompt about an existing destination file
-v, --verbose	:	explain what is being done
--help	:	display this help and exit
--version	:	output version information and exit

10. RM

rm - remove files or directories

rm [OPTION]... FILE...

DESCRIPTION

rm removes each specified file. By default, it does not remove directories. If a file is unwritable, the standard input is a tty, and the -f or --force option is not given, rm prompts the user for whether to remove the file. If the response does not begin with `y' or `Y', the file is skipped.

OPTIONS

Remove (unlink) the FILE(s).

-d, --directory	:	unlink FILE, even if it is a non-empty directory (super-user only)
-i, --interactive	:	prompt before any removal
-r, -R, --recursive	:	remove the contents of directories recursively
-v, --verbose	:	explain what is being done
--help	:	display this help and exit
--version	:	output version information and exit

11. RMDIR

rmdir - remove empty directories

rmdir [OPTION]... DIRECTORY...

DESCRIPTION

Remove the DIRECTORY(ies), if they are empty.

-p, --parents	:	remove DIRECTORY, then try to remove each directory component of that path name. E.g., `rmdir -p a/b/c' is similar to `rmdir a/b/c a/b a'.
-v, --verbose	:	output a diagnostic for every directory processed
--help	:	display this help and exit
--version	:	output version information and exit

12. DATE

date - print or set the system date and time

date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION

Display the current time in the given FORMAT, or set the system date.

-d, --date=STRING	:	display time described by STRING, not `now'
-s, --set=STRING	:	set time described by STRING
-u, --utc, --universal	:	print or set Coordinated Universal Time
--help	:	display this help
--version	:	output version information and exit

FORMAT controls the output. The only valid option for the second form specifies Coordinated Universal Time. Interpreted sequences are:

%%	:	a literal %
%a	:	locale's abbreviated weekday name (Sun..Sat)
%A	:	locale's full weekday name, variable length (Sunday..Saturday)
%b	:	locale's abbreviated month name (Jan..Dec)
%B	:	locale's full month name, variable length (January..December)
%c	:	locale's date and time (Sat Nov 04 12:02:33 EST 1989)
%d	:	day of month (01..31)
%D	:	date (mm/dd/yy)
%e	:	day of month, blank padded (1..31)
%h	:	same as %b
%H	:	hour (00..23)
%I	:	hour (01..12)
%j	:	day of year (001..366)
%k	:	hour (0..23)
%l	:	hour (1..12)
%m	:	month (01..12)
%M	:	minute (00..59)
%n	:	a newline
%p	:	locale's AM or PM
%r	:	time, 12-hour (hh:mm:ss [AP]M)
%s	:	seconds since `00:00:00 1970-01-01 UTC' (a GNU extension)
%S	:	second (00..60)
%t	:	a horizontal tab
%T	:	time, 24-hour (hh:mm:ss)
%U	:	week number of year with Sunday as first day of week (00..53)
%V	:	week number of year with Monday as first day of week (01..53)
%w	:	day of week (0..6); 0 represents Sunday
%W	:	week number of year with Monday as first day of week (00..53)
%x	:	locale's date representation (mm/dd/yy)
%X	:	locale's time representation (%H:%M:%S)
%y	:	last two digits of year (00..99)
%Y	:	year (1970...)

By default, date pads numeric fields with zeroes. GNU date recognizes the following modifiers between `%' and a numeric directive.

`-' (Hyphen) do not pad the field `_' (underscore) pad the field with spaces

13. FIND

find - search for files in a directory hierarchy

find [option]

DESCRIPTION

Find searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence until the outcome is known (the left hand side is false for and operations, true for or), at which point find moves on to the next file name.

The first argument that begins with `-', `(', `)', `,', or `!' is taken to be the beginning of the expression; any arguments before it are paths to search, and any arguments after it are the rest of the expression. If no paths are given, the current directory is used. If no expression is given, the expression `-print' is used.
find exits with status 0 if all files are processed successfully, greater than 0 if errors occur.

OPTIONS

All options always return true. They always take effect, rather than being processed only when their place in the expression is reached. Therefore, for clarity, it is best to place them at the beginning of the expression.

-depth	:	Process each directory's contents before the directory itself.
-help, --help	:	Print a summary of the command-line usage of find and exit.
-maxdepth levels	:	Descend at most levels (a non-negative integer) levels of directories below the command line arguments. `-maxdepth 0' means only apply the tests and actions to the command line arguments.
-version, --version	:	Print the find version number and exit.
-gid n	:	File's numeric group ID is n.
-group gname	:	File belongs to group gname (numeric group ID allowed).
-iname pattern	:	Like -name, but the match is case insensitive. For example, the patterns `fo*' and `F??' match the file names `Foo', `FOO', `foo', `fOo', etc.
-ipath pattern	:	Like -path, but the match is case insensitive.
-name pattern	:	Base of file name (the path with the leading directories removed) matches shell pattern pattern. The metacharacters (`*', `?', and `[]') do not match a `.' at the start of the base name.
-path pattern	:	File name matches shell pattern pattern. The metacharacters do not treat `/' or `.' pecially; so, for example,find . -path './sr*sc' will print an entry for a directory called './src/misc' (if one exists).
-type c	:	File is of type c:
c	:	character (unbuffered) special
d	:	directory
f	:	regular file
l	:	symbolic link
-uid n	:	File's numeric user ID is n.
-user uname	:	File is owned by user uname (numeric user ID allowed).

14. CMP

compare two files or byte ranges

cmp [-clsv] [-i NUM] [--help] [--print-chars] [--ignore-initial=NUM] [--verbose] [--quiet] [--silent] [--version] -I FILE1 [FILE2 [RANGE1 [RANGE2]]]

DESCRIPTION

The cmp utility compares two files of any type and writes the results to the standard output. By default, cmp is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported.

In the output, bytes and lines are numbered beginning with one; however, range inputs are zero-based; see below for details. A filename of - represents standard input.

The following options are available:

-c, --print-chars	:	Output the differing bytes as characters, rather than as octal numbers. Non-printable characters will be shown in form.
-i NUM, --ignore-initial=NUM:		Ignore NUM initial characters from each file. This is a synonym for specifying NUM NUM as the two RANGE arguments.
-l, --verbose	:	Print the byte number (decimal) and the differing byte values (octal) for each difference.
-s, --quiet, --silent	:	Print nothing for differing files; return exit status only.
-v, --version	:	Print the diffutils version number.

BYTE RANGES

The two optional arguments RANGE1 and RANGE2 represent byte ranges to compare within the files. Each range can be expressed in several ways:

M+N
Skip M bytes at the beginning of the input, then compare a maximum of N bytes.
M-N, M,N
Skip M bytes at the beginning of the input, and read between bytes M and N, which are both zero-based.
In either case, both M and N are optional and default to beginning and end of file, respectively. In addition, they can be expressed in decimal, octal (0NNN) or hexadecimal (0xNNN) form.

DIAGNOSTICS
The cmp utility exits with one of the following values:

0	:	The files or byte ranges are identical.
1	:	The files or byte ranges are different; this includes the case where one file or range is identical to the first part of the other. In the latter case, if -s has not been specified, cmp writes to standard output that EOF was reached in the shorter file.
>1	:	An error occurred.

15. CHMOD

change file access permissions

chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...

DESCRIPTION

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.

The format of a symbolic mode is `[ugoa...][[+-=][rwx]...][,...]'. Multiple symbolic operations can be given, separated by commas.

A combination of the letters `ugoa' controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users (a). If none of these are given, the effect is as if `a' were given.

The operator `+' causes the permissions selected to be added to the existing permissions of each file; `-' causes them to be removed; and `=' causes them to be the only permissions that the file has.

The letters `rwx' select the new permissions for the affected users: read (r), write (w), execute (or access for directories) (x).

A numeric mode is from one to three octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects permissions for the user who owns the

file: read (4), write (2), and execute (1); the second permissions for other users in the file's group, with the same values; and the third for other users not in the file's group, with the same values.

chmod never changes the permissions of symbolic links; the chmod system call cannot change their permissions. This is not a problem since the permissions of symbolic links are never used. However, for each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

OPTIONS

Change the mode of each FILE to MODE.

-c, --changes	:	like verbose but report only when a change is made
-v, --verbose	:	output a diagnostic for every file processed
-R, --recursive	:	change files and directories recursively
--help	:	display this help and exit
--version	:	output version information and exit

16. CHOWN

change file owner and group

chown [OPTION]... OWNER[:[GROUP]] FILE...
chown [OPTION]... :GROUP FILE...

DESCRIPTION

chown changes the user and/or group ownership of each given file, according to its first non-option argument, which is interpreted as follows. If only a user name (or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed. If the user name is followed by a colon or dot and a group name (or numeric group ID), with no spaces between them, the group ownership of the files is changed as well. If a colon or dot but no group name follows the user name, that user is made the owner of the files and the group of the files is changed to that user's login group. If the colon or dot and group are given, but the user name is omitted, only the group of the files is changed; in this case, chown performs the same function as chgrp.

OPTIONS

Change the owner and/or group of each FILE to OWNER and/or GROUP.

-c, --changes	:	like verbose but report only when a change is made
-R, --recursive	:	operate on files and directories recursively
-v, --verbose	:	output a diagnostic for every file processed
--help	:	display this help and exit
--version	:	output version information and exit

Owner is unchanged if missing. Group is unchanged if missing, but changed to login group if implied by a `:'. OWNER and GROUP may be numeric as well as symbolic.

17. CHGRP

change group ownership

chgrp [OPTION]... GROUP FILE...

DESCRIPTION

Change the group membership of each FILE to GROUP.

-c, --changes	:	like verbose but report only when a change is made
-R, --recursive	:	operate on files and directories recursively
-v, --verbose	:	output a diagnostic for every file processed
--help	:	display this help and exit
--version	:	output version information and exit

18. KILL

Terminate a process

kill pid ...

DESCRIPTION

The command kill sends the specified signal to the specified process or process group.

Pid : Specify the list of processes that kill should signal.

19. TTY

tty - print the file name of the terminal connected to standard input

tty [OPTION]...

DESCRIPTION

Print the file name of the terminal connected to standard input.

--help	:	display this help and exit
--version	:	output version information and exit

20. WALL

wall -- send a message to everybody's terminal.

wall [message]

DESCRIPTION

Wall sends a message to everybody logged in with their mesg(1) permission set to yes. The message can be given as an argument to wall, or it can be sent to wall's standard input. When using the standard input from a terminal, the message should be terminated with the EOF key (usually Control-D).

The length of the message is limited to 20 lines.

NOTES: There is an undocumented (well not anymore..) option, "-n", that supresses the banner printed by wall . This is for usage by rwalld (8). Wall will not allow you to use that flag if wall is installed set-group-id and the user executing wall is not root.

21. WHO

who - show who is logged on

who [OPTION]

DESCRIPTION

-H, --heading	:	print line of column headings
-m	:	only hostname and user associated with stdin
-q, --count	:	all login names and number of users logged on
--help	:	display this help and exit
--version	:	output version information and exit

22. MESG

mesg - control write access to your terminal

mesg [y|n]

DESCRIPTION

Mesg controls the access to your terminal by others. It's typically used to allow or disallow other users to write to your terminal (see write(1)).

OPTIONS

y	:	Allow write access to your terminal.
n	:	Disallow write access to your terminal.

If no option is given, mesg prints out the current access state of your terminal. Mesg assumes that it's standard input is connected to your terminal. That also means that if you are logged in multiple times, you can get/set the mesg status of other sessions by using redirection. For example "mesg n < /dev/pts/46".

23. WRITE

write - send a message to another user

write user [ttyname]

DESCRIPTION

Write allows you to communicate with other users, by copying lines from your terminal to theirs.
When you run the write command, the user you are writing to gets a message of the form:
Message from yourname@yourhost on yourtty at hh:mm ...

Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well.

When you are done, type an end-of-file or interrupt character. The other user will see the message EOF indicating that the conversation is over.
You can prevent people (other than the super-user) from writing to you with the mesg(1) command. Some commands, for example nroff(1) and pr(1), may disallow writing automatically, so that your output isn't overwritten.

If the user you want to write to is logged in on more than one terminal, you can specify which terminal to write to by specifying the terminal name as the second operand to the write command. Alternatively, you can let

write select one of the terminals - it will pick the one with the shortest idle time. This is so that if the user is logged in at work and also dialed up from home, the message will go to the right place.

The traditional protocol for writing to someone is that the string `-o', either at the end of a line or on a line by itself, means that it's the other person's turn to talk. The string `oo' means that the person believes the conversation to be over.

24. VIM / VI

vim - Vi IMproved, a programmers text editor

vim [options] [file ..]

DESCRIPTION

Vim is a text editor that is upwards compatible to Vi. It can be used to edit any ASCII text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between Vim and Vi.

While running Vim a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.
Most often Vim is started to edit a single file with the command

vim <file>

More generally Vim is started with:

vim [options] [file list]

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

file : A list of filenames. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. You can get to the other files with the ": next" Command. To edit a file that starts with a dash, precede the file list with "--".

Vim behaves differently, depending on the name of the command (the executable may still be the same file).

Vim : The "normal" way, everything is default.
Ex : Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "-e" argument.
View : Start in read-only mode. You will be protected from writing the files. Can also be done With the "-R" argument.
gvim gview : The GUI version. Starts a new window. Can also be done with the "-g" argument.
rvim rview rgvim rgview Like the above, but with restrictions. It will not be possible to start shell commands, or suspend Vim. Can also be done with the "-Z" argument.

OPTIONS

The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash.

-c {command} : {command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be

		enclosed in double quotes (this depends on the shell that is used).
--cmd {command}	:	Like using "-c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "-c" commands.
-b	:	Binary mode. A few options will be set that makes it possible to edit a binary or executable file.
-C	:	Compatible. Set the 'compatible' option. This will make Vim behave mostly like Vi, even though a .vimrc file exists.
-d	:	Start in diff mode. There should be two or three file name arguments. Vim will open all the files and show differences between them. Works like vimdiff(1).
-d {device}	:	Open {device} for use as a terminal. Only on the Amiga. Example: "-d con:20/30/600/150".
-e	:	Start Vim in Ex mode, just like the executable was called "ex".
-f	:	Foreground. For the GUI version, Vim will not fork and detach from the shell it was started in. On the Amiga, Vim is not restarted to open a new window. This option should be used when Vim is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and ":!" commands will not work.
-o[N]	:	Open N windows. When N is omitted, open one window for each file.
-R	:	Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from accidently overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The -R option also implies the -n option (see below). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'".
-V	:	Verbose. Give messages about which files are sourced and for reading and writing a viminfo file.
-v	:	Start Vim in Vi mode, just like the executable was called "vi". This only has effect when the executable is called "ex".
--help	:	Give a help message and exit, just like "-h".
--version	:	Print version information and exit.

ON-LINE HELP Type ":help" in Vim to get started. Type ":help subject" to get help on a specific subject. For example: ":help ZZ" to get help for the "ZZ" command. Use <Tab> and CTRL-D to complete subjects (":help cmdline-completion"). Tags are present to jump from one place to another (sort of hypertext links, see ":help").

The 'vi' Modes

These are three basic modes of operation in 'vi'. The three modes are.

- **Command Mode:** The 'command mode' is one in which user is able to execute commands.
- **Insert or Input Mode:** The 'Input Mode' or the Insert mode, the user can enter the data or edit the data.
- **The Last Line Mode:** The last line mode is one in which the user is able to enter special commands, which typically start with, the '**:**' (colon).

'vi' Commands:-

Command to Insert Text

Command	Purpose
i	Invokes insert mode and inserts before character
l	Same as 'i' but insertion is at beginning of line
o	Opens black line before the current line

O	Same as ‘o’ but puts black line after current line
a	Same as ‘i’ but appends after cursor
A	Same as ‘a’ but appends at end of line

Navigating the Cursor

Command	**Purpose**
k	Same as the ‘up’ arrow key
h	Same as the ‘left’ arrow key
l	Same as the ‘right’ arrow key
j	Same as the ‘down’ arrow key
w	Moves forward by a word
‘n’w	Moves forward by ‘n’ number of words
e	Moves to the last character of this word or that of the next
‘n’e	Moves to the last character of n^{th} next word
b	Takes the cursor back by a word
‘n’b	Takes the cursor back by ‘n’ number of words
0(Zero) or ^	Takes the cursor to the beginning of the line
$	Takes the cursor to the end of the line
‘n’G	Takes the cursor to the ‘n’ th line
L	Takes the cursor to the last line in the screen.
H	Takes the cursor to the first line in the screen
M	Takes the cursor to the middle of the screen
{	Moves the cursor to the beginning of the previous paragraph
}	Moves the cursor to the beginning of the next paragraph
(	Moves the cursor to the beginning of the previous sentence
)	Moves the cursor to the beginning of the next sentence

Commands for scrolling

Command	**Purpose**
CTRL + f	Scrolls a screen forward
CTRL + b	Scrolls a screen backward
CTRL + u	Scrolls half a screen backward
CTRL + d	Scrolls half a screen forward
CTRL + L	Refreshes the screen

Commands for Deleting Text

Command	**Purpose**
x	Deletes one character at the cursor position
‘n’x	Deletes the next ‘n’ character from the cursor position
dw	Deletes one word at the cursor position
‘n’dw	Deletes ‘n’ word at the cursor position
d0(zero)	Deletes from the beginning of the line till the cursor position
dd	Delete from the cursor position till the end of the line
‘n’dd	Delete the next ‘n’ lines from the cursor position
u	Undoing a change

Commands to Replace Existing Text

Command	**Purpose**
r	Replace s single character at the current position
R	Invoke insert mode and go on replacing till <Esc> is pressed
s	Deletes the current character and inserts characters until <Esc> is pressed
S	Removes all the characters in the current line and allows the user to enter data

Commands for Saving and Existing

Command	Purpose
:q	Quit vi
:q!	Quit vi without saving changes
:w	Save and continue working in vi
:wq	Save the changes and exit from vi
:x	Save and exit from vi
ZZ	Save the changes and exit from vi

Commands to Copy Text

Command	Purpose
yy	Copy a line into temporary storage
'n'yy	Copy 'n' number of lines into temporary storage
dd	Move a line into temporary storage
'n'dd	Move 'n' lines into temporary storage
p	Move the text from temporary storage to the line below the current position of the cursor
P	Move the text from temporary storage to the line above the current position of the cursor

Commands for Pattern Searching

Command	Purpose
/g	Search for a pattern 'g' forwards
?g	Search for pattern 'g' backwards
/ or n	Repeat the last search forward
? or N	Repeat the last search backward

25. MAIL

mail - send and receive mail

mail [-iInv] [-s subject] [-c cc-addr] [-b bcc-addr] to-addr...
mail [-iInNv -f] [name]
mail [-iInNv [-u user]]

INTRODUCTION

Mail is an intelligent mail processing system, which has a command syntax reminiscent of ed1 with lines replaced by messages.

-v	:	Verbose mode. The details of delivery are displayed on the user's terminal.
-I	:	Forces mail to run in interactive mode even when input isn't a terminal. In particular, the `~ ' special character when sending mail is only active in interactive mode.
-n	:	Inhibits reading /etc/mail.rc upon startup.
-N	:	Inhibits the initial display of message headers when reading mail or editing a mail folder.
-s	:	Specify subject on command line (only the first argument after the -s flag is used as a subject; be careful to quote subjects containing spaces.)
-c	:	Send carbon copies to list of users.
-b	:	Send blind carbon copies to list List should be a comma-separated list of names.
-f	:	Read in the contents of your mbox (or the specified file) for processing; when you quit mail writes undeleted messages back to this file.
-u	:	Is equivalent to: mail -f /var/spool/mail/user

Sending mail To send a message to one or more people, mail can be invoked with arguments which are the names of people to whom the mail will be sent. You are then expected to type in your message, followed by an `control-D ' at the beginning of a line. The section below Replying to or originating mail describes some features of mail available to help you compose your letter.

Reading mail In normal usage mail is given no arguments and checks your mail out of the post office, then prints out a one line header of each message found. The current message is initially the first message (numbered 1) and can be printed using the print command (which can be abbreviated `p ' ) . You can move among the messages much as you move between lines in ed1, with the commands `+ ' and `- ' moving backwards and forwards, and simple numbers.

Disposing of mail. After examining a message you can delete `d ' ) the message or reply `r ') to it. Deletion causes the mail program to forget about the message. This is not irreversible; the message can be undeleted `u ' ) by giving its number, or the mail session can be aborted by giving the exit `x ') command. Deleted messages will, however, usually disappear never to be seen again.

Specifying messages Commands such as print and delete can be given a list of message numbers as arguments to apply to a number of messages at once. Thus ``delete 1 2 '' deletes messages 1 and 2, while ``delete 1-5 '' deletes messages 1 through 5. The special name `* ' addresses all messages, and `$ ' addresses the last message; thus the command top which prints the first few lines of a message could be used in ``top * '' to print the first few lines of all messages.

Replying to or originating mail. You can use the reply command to set up a response to a message, sending it back to the person who it was from. Text you then type in, up to an end-of-file, defines the contents of the message. While you are composing a message, mail treats lines beginning with the character `~ ' specially. For instance, typing `~m ' (alone on a line) will place a copy of the current message into the response right shifting it by a tab stop (see indent prefix variable, below). Other escapes will set up subject fields, add and delete recipients to the message and allow you to escape to an editor to revise the message or to a shell to run some commands. (These options are given in the summary below.)

Ending a mail processing session. You can end a mail session with the quit `q ') command. Messages, which have been examined go to your mbox file unless they have been deleted in which case they are discarded. Unexamined messages go back to the post office. (See the -f option above).

Personal and system wide distribution lists. It is also possible to create a personal distribution lists so that, for instance, you can send mail to ``cohorts '' and have it go to a group of people. Such lists can be defined by placing a line like alias cohorts bill ozalp jkf mark kridle@ucbcory in the file .mailrc in your home directory. The current list of such aliases can be displayed with the alias command in mail System wide distribution lists can be created by editing /etc/aliases see aliases(5) and sendmail(8); these are kept in a different syntax. In mail you send, personal aliases will be expanded in mail sent to others so that they will be able to reply to the recipients. System wide aliases are not expanded when the mail is sent, but any reply returned to the machine
will have the system wide alias expanded as all mail goes through send mail.

FILE SYSTEMS

---

filesystems - Linux filesystem types: minix, ext, ext2, ext3, xia, msdos, umsdos, vfat, proc, nfs, iso9660, hpfs, sysv, smb, ncpfs

DESCRIPTION

When, as is customary, the proc filesystem is mounted on /proc, you can find in the file /proc/filesystems which filesystems your kernel currently supports. If you need a currently unsupported one, insert the corresponding module or recompile the kernel. In order to use a filesystem, you have to mount it, see mount(8) for the mount command, and for the available mount options.

Below a short description of a few of the available filesystems.

minix	is the filesystem used in the Minix operating system, the first to run under Linux. It has a number of shortcomings: a 64MB partition size limit, short filenames, a single time stamp, etc. It remains useful for floppies and RAM disks.
ext	is an elaborate extension of the minix filesystem. It has been completely superseded by the second version of the extended filesystem (ext2) and has been removed from the kernel (in 2.1.21).
ext2	is the high performance disk filesystem used by Linux for fixed disks as well as removable media. The second extended filesystem was designed as an extension of the extended file system (ext). ext2 offers the best performance (in terms of speed and CPU usage) of the filesystems supported under Linux.
ext3	is a journaling version of the ext2 filesystem. It is easy to switch back and forth between ext2 and ext3.
xiafs	was designed and implemented to be a stable, safe filesystem by extending the Minix filesystem code. It provides the basic most requested features without undue complexity. The xia filesystem is no longer actively developed or maintained. It was removed from the kernel in 2.1.21.
msdos	is the filesystem used by DOS, Windows, and some OS/2 computers. msdos filenames can be no longer than 8 characters, followed by an optional period and 3 character extension.
umsdos	is an extended DOS filesystem used by Linux. It adds capability for long filenames, UID/GID, POSIX permissions, and special files (devices, named pipes, etc.) under the DOS filesystem, without sacrificing compatibility with DOS.
vfat	is an extended DOS filesystem used by Microsoft Windows95 and Windows NT. VFAT adds the capability to use long filenames under the MSDOS filesystem.
proc	is a pseudo-filesystem which is used as an interface to kernel data structures rather than reading and interpreting /dev/kmem. In particular, its files do not take disk space.
iso9660	is a CD-ROM filesystem type conforming to the ISO 9660 standard.
high sierra	Linux supports High Sierra, the precursor to the ISO 9660 standard for CD-ROM filesystems. It is automatically recognized within the iso9660 filesystem support under Linux.
rock ridge	Linux also supports the System Use Sharing Protocol records specified by the Rock Ridge Interchange Protocol. They are used to further describe the files in the iso9660 filesystem to a UNIX host, and provide information such as long filenames, UID/GID, POSIX permissions, and devices. It is automatically recognized within the iso9660 filesystem support under Linux.
hpfs	is the High Performance Filesystem, used in OS/2. This filesystem is read-only under Linux due to the lack of available documentation.
sysv	is an implementation of the SystemV/Coherent filesystem for Linux. It implements all of Xenix FS, SystemV/386 FS, and Coherent FS.
nfs	is the network filesystem used to access disks located on remote computers.
smb	is a network filesystem that supports the SMB protocol, used by Windows for Workgroups, Windows NT, and Lan Manager. To use smb fs.
ncpfs	is a network filesystem that supports the NCP protocol, used by Novell NetWare.

su Command

The su command, or substitute user command, allows you to run a command as any user on your system. Found in the /bin directory, su has seven different command-line options. Several of the most common are covered here. Although you'll most likely use su to become root, this command can be handy if you want to become another user and troubleshoot problems with email or printing.

Unless you specify your own username, you'll run the new shell as the root operator. A better approach is

$ su -s /bin/ksh yourusername

Finally, you can also use the su command to execute a single command when you use the -c command-line option. This can be handy to perform tasks only permitted for the root operator, such as mounting a flash memory card:

su -c "mount -t vfat /dev/hde1 /mnt/flash"
Password:

Fdisk

fdisk—Partition table manipulator for Linux.

fdisk [-l] [-v] [-s partition] [device]

DESCRIPTION

fdisk is a menu-driven program for manipulation of the hard disk partition table. The device is usually one of the following:
/dev/hda
/dev/hdb
/dev/sda
/dev/sdb

The partition is a device name followed by a partition number. For example, /dev/hda1 is the first partition on the first hard disk in the system. If possible, fdisk will obtain the disk geometry automatically. This is not necessarily the physical disk geometry but is the disk geometry that MS-DOS uses for the partition table. If fdisk warns you that you need to set the disk geometry, please believe this statement and set the geometry. This should only be necessary with certain SCSI host adapters (the drivers for which are rapidly being modified to provide geometry information automatically).
Whenever a partition table is printed, a consistency check is performed on the partition table entries. This check verifies that the physical and logical start and end points are identical and that the partition starts and ends on a cylinder boundary (except for the first partition).

OPTIONS

-v Prints version number of fdisk program.
-l Lists the partition tables for /dev/hda, /dev/hdb, /dev/sda, /dev/sdb, /dev/sdc, /dev/sdd, /dev/sde, /dev/sdf, /dev/sdg, and /dev/sdh and then exits.

-s partition If the partition is not a DOS partition (the partition ID is greater than 10), then the size of that partition is printed on the standard output. This value is usually used as an argument to the mkfs(8) program to specify the size of the partition that will be formatted.

Getting Filesystem Statistics with the df Command

The df (free disk space) command, found in the /bin directory, will gather and summarize some important statistics about all currently mounted filesystems. The df command is easy to use:

df

Filesystem	1024-blocks	Used	Available	Capacity	Mountedon
/dev/hda3	497699	443871	28124	94%	/
/dev/hda1	509856	469632	40224	92%	/mnt/dos
/dev/hdc1	3868	2596	1272	67%	/mnt/flash
/dev/hdb	644324	644324	0	100%	/mnt/cdrom

This output shows four different filesystems on three different devices mounted under Linux. The first is the root partition at the / directory on /dev/hda3; the second is a DOS partition under /mnt/dos on /dev/hda1; the third is a flashcard under /mnt/flash on
/dev/hdc1; and the fourth is a CD-ROM, mounted under /mnt/cdrom on /dev/hdb. The
df command also lists the size of the storage device, how much has been used, how much is available, and the current capacity of the device. Notice that the CD-ROM has no space left. This is because it is mounted read-only, meaning you can't save or delete files on this device.

Mount

The mount command is a powerful tool used to mount directories and devices on a Linux system. The simplest application of the mount command is to use it with no options or arguments.

Option	Description
-a	Mount all filesystems mentioned in fstab of a given type.
-h	Provides the help file information.
-L label	Mount a partition that has the specified label.
-n	Mount without writing information in /etc/mtab.
-r	Mount the file system as read-only.
-t vfstype	Used to indicate file system type.
-u uuid	Mount the partition that has the specified uuid and requires the file /proc/partitions, available since Linux 2.1.116.
-V	Display mount version.
-v	Use verbose mode.
-w	Mount the file system read/write. This is the default.
-o argument	Options are specified with a -o flag followed by a comma separated string of options. More on this option in the -o option section.

These options are used to mount a device. The following is a common command, which is used to mount the diskette drive on a Linux system.

```
$ mount -v /mnt/floppy
/mnt/floppy on /mnt/floppy type supermount (rw,fs=vfat,dev=/dev/fd0)
```

Notice that the file system type is listed and can be assigned by the -t option.

Arguments for the -o Option

Argument	Description
async	All I/O to the file system should be done asynchronously.
atime	Update inode access time for each access. This is the default.
auto	Can be mounted with the -a option.
defaults	Use default options: rw, suid, dev, exec, auto, nouser, and async.
dev	Interpret character or block special devices on the file system.
exec	Permit the execution of binaries.
noatime	Do not update inode access times on this file system.
noauto	Can only be mounted explicitly.
nodev	Do not interpret character or block special devices on the file system.
noexec	Do not allow execution of any binaries on the mounted file system.
nosuid	Do not allow set-user-identifier or set-group-identifier bits to take effect.
nouser	Forbid a non-root user to mount the file system. This is the default.
remount	Attempt to remount an already-mounted file system.
ro	Mount the file system read-only.
rw	Mount the file system read-write.
suid	Allow set-user-identifier or set-group-identifier bits to take effect.
sync	All I/O to the file system should be done synchronously.
user	Allow an ordinary user to mount the file system.

Automatically Mounting File System : fstab file

The /etc/fstab file is consulted whenever the mount command is started. The table lists the different file systems that are located on each partition, and gives options on how they are loaded. A typical /etc/fstab looks something like this:

#Device	**Mountpoint**	**FileSystem**	**Option**	**Dump**	**fsckorder**
/dev/hda1	/dos	msdos	defaults	0	3
/dev/hda5	/boot	ext2	defaults	1	2
/dev/hda6	/	ext2	defaults	1	1
/dev/hda7	swap	swap	defaults	0	0
/dev/fd0	/mnt/floppy	ext2	user,noauto	0	0
/dev/hdc	/cdrom	iso9660	user,ro,noauto	0	0
none	/proc	proc	defaults	0	0
none	/dev/pts	devpts	gid=5,mode=620	0	0

An entry in a fstab file contains several fields, each separated by a space or tab. The first field is the name of the file system to be mounted. This usually begins with /dev, such as /dev/hda3 for the third hard disk partition. The next field is the directory in your file structure where you want the file system on this device to be attached. The third field is the type of the file system being mounted. The type for a standard Linux hard disk partition is

ext3. the next example shows as entry for the main Linux hard disk partition. It is mounted at the root directory '/' and has a file type of ext3.

The field after the file system type lists the different options for mounting the file system. There is a default set of options that you can specify by simply entering **defaults**. You can list option next to each other separated by comma (no spaces). The defaults options that a device is read/write, asynchronous, block, that ordinary users cannot mount on it, and programs can be executed on it.

The last two fields consist of an integer value. The first one is used by the dump command to determine if a file system needs to be dumped, backing up the file system. The last one is used by fsck to see if a file system should be checked and in what order. If the field has a value of 1, it indicates a boot partition. The 0 value means that the fsck does not have to check the file system.

Umount

The opposite of mounting a drive is to unmount it, so that it cannot be accessed.
Use the umount command to unmount a file system. If a floppy diskette drive is mounted, unmount the diskette before removing it. To perform this task, use the following umount command :

```
$ umount -v /mnt/floppy
/mnt/floppy umounted
```

Option	Description
-a	Un-mount all file systems mentioned in /etc/mtab.
-h	Provides the help file information.
-n	Un-mount without writing information in /etc/mtab.
-r	In case umount fails attempt to mount the file system as read-only.
-t vfstype	Used to indicate file system type.
-V	Display umount version.
-v	Use verbose mode.

The du command

The du command is used to summarize disk usage of each file and recursively for directories. This command provides an extensive list of all the files and their usage of storage space. You may use the du command to determine the space used on a diskette as shown in the following:

```
$ ls
picture.bmp*
$ du
769 .
```

The picture file uses 769 kilobytes of the space on the diskette. The available options for the du command are shown in Table 9-18.

Option	Description
-a --all	Write counts for all files, not just directories.
-b --bytes	Print size in byte.
-c --total	Provide a grand total.
-D --dereference-args	De-reference PATHs when symbolic link.

-h --human-readable	Print sizes in human readable format, such as M for MB.
-H --si	Print sizes in human readable format but uses powers of 1000 instead of 1024, which the -h option uses.
-l --count-links	Count sizes many times if hard linked.
-L --dereference	De-reference all symbolic links.
-s --summarize	Display only a total for each argument.
--max-depth=N	Print the total for a directory (or file, with --all) only if it is N or fewer levels below the command line argument.
--help	Provide help with the program.
--version	Provides version information about the program.

The df comand

The df command is similar to the du command but provides information about the amount of disk space available on the file system. The following is an example of the information provided by the df command:

$ df –m

Filesystem	1M-blocks	Used	Available	Use%	Mounted on
/dev/hda1	248	77	171	31%	/
/dev/hda6	913	687	226	75%	/usr
/dev/hda7	453	59	394	13%	/var

Option	Description
-a --all	Include file systems having 0 blocks.
-h --human-readable	Print sizes in human readable format, such as M for MB.
-H --si	Print sizes in human readable format but uses powers of 1000 instead of 1024, which the -h option uses.
-i	List inode information instead of block usage.
-m --megabytes	Sets block-size=1048576.
--no-sync	Do not invoke sync before getting usage info, the default.
-p --portability	Use the POSIX output format.
-t --type=TYPE	Limit the listing to file systems of type TYPE.
-T --print-type	Print the file system type.
-v	Ignored.
--help	Provide help with the program.
--version	Provides version information about the program.

more

more—File perusal filter for crt viewing. more is a filter for paging through text one screenful at a time. This version is especially primitive.

more [-dps] [-num] [+/ pattern] [+ linenum]

OPTIONS

-num	This option specifies an integer that is the screen size (in lines).
-d	more will prompt the user with the message [Press space to continue, 'q' to quit.] and will display [Press 'h' for instructions.] instead of ringing the bell when an illegal key is pressed.
-p	Do not scroll. Instead, clear the whole screen and then display the text.
-s	Squeeze multiple blank lines into one.
+/	The +/ option specifies a string that will be searched for before each file is displayed.
+num	Start at line number.

Backup (File Comparisons)

tar

The tar command, short for **t**ape **ar**chive, is one of the oldest Unix commands for creating archive files for storage on magnetic tape. The tar command isn't limited for use on tape devices; it is commonly used to create any type of file archive and to save it to any type of device.

The tar command does not compress data; it only creates file archives. For backup use, you should use the hardware compression that's built into your storage unit. If you want to compress files with tar, you can use it in conjunction with the compress or gzip commands.

The format of the tar command is:

tar [options] (file1) (file2) ...

You can use several options and arguments with the tar command:

- c Tells tar to create a new archive file
- v Verbose mode; this will print each filename as it is archived
- f Lets you specify a filename for the archive
- x Tells tar to extract files from an archive
- p Tells tar to keep permissions on files when extracted from an archive

To back up the entire /home directory to a tape device, use the following command:

tar -cvf /dev/tape /home

To extract the same file into the current directory, use the following command:

tar -xvfp /dev/tape

To see the contents of a tar archive on tape, use the following command:

tar -tvf /dev/tape

To restore a single file from tape, use the following command:

tar -xvfp /dev/tape /home/file

dump and restore

Use the dump command to dump entire Linux file systems to the output device of your choice, which can be a disk file or a tape device.

Here are some of the command options that can be used with the dump command:

- [0-9]: This option sets the dump level for the current operation. A dump level of 0 indicates a full backup. Any level higher than 0 indicates an incremental backup, so dump copies all new or modified files since the last dump of the same or lower level.
- f: This option indicates the file or device that the command is writing to.

To dump the entire /home partition to a tape device, use the following command:

```
dump -f /dev/tape -0 /home
```

Use the **restore** command to restore files that were backed up by using the dump command. The dump command saves the directory structure of the files that were backed up, so when you invoke the restore command, you can navigate up and down the directory tree to choose the files to restore.

You must properly mount the tape, and fast-forward or rewind it to the archive that contains the file that you want to restore.
When you type the restore command, you receive a prompt similar to the following:

```
restore>
```

Type ? for a list of commands; the most used commands are add and extract. You can also use cd and ls to find the files that you want. Then use the following command:

```
restore> add filename
```

Continue to add all the files that you want to restore, and then use the following command:

```
restore> extract
```

This command will then restore the files into your current directory, so before invoking the restore command, you should navigate to the directory where you want to place the recovered files.

Compressed archive

A compressed archive consists of all the files that are part of the update, tied together in a single file called a tar archive.

One problem with tar is that it doesn't actually compress the files in the archive; therefore, they can often be very large in size, which makes them more difficult to download over slow Internet connections, and impossible to fit on a floppy disk. To compress the archive, use the gzip utility. This utility compresses the data in a file into the smallest possible form. When used in conjunction with the tar utility, you can create a collection of files, and then compress them so that you only need one small file.

The resulting file has a .tar.gz extension, indicating that it is a tar file that has been compressed with gzip. To extract the file, use the gunzip command to unzip the file:

gunzip packagename.tar.gz

After the command completes, the file will be unzipped, and the remaining product is the tar archive file. Then, you can use the tar command to extract the files from the archive:

tar -xvf packagename.tar

The tar utility also comes with the capability of unzipping the archive while extracting files at the same time. To do this, use the -z switch:

tar -zxvf packagename.tar.gz

Performing the gunzip and tar extract commands together is a lot easier than using the two separate commands to extract and unzip the files.

Gzip and Gunzip

To use gzip to compress a _le, type the following command at a shell prompt:

gzip filename

The file will be compressed and saved as filename.gz.

To expand the compressed _le, type the following command:

gunzip filename.gz

The filename.gz is deleted and replaced with filename.

You can use gzip to compress multiple _les and directories at the same time by listing them with a space between each one:

gzip -r filename.gz file1 file2 file3 /usr/work/school

The above command compresses file1, file2, file3, and the contents of the /usr/work/school directory (assuming this directory exists) and places them in a file named filename.gz.

fsck

fsck—Check and repair a Linux filesystem.

fsck [-AVRTN][-s][-t fstype][fs-options] filesys [...]

DESCRIPTION

fsck is used to check and optionally repair a Linux filesystem. filesys is either the device name (such as /dev/hda1 or /dev/ sdb2) or the mount point (such as /, /usr, or /home) for the filesystem. If this fsck has several filesystems on different physical disk drives to check, this fsck will try to run them in parallel. This reduces the total amount time it takes to check all of the filesystems because fsck takes advantage of the parallelism of multiple disk spindles.

The exit code returned by fsck is the sum of the following conditions:

0 No errors
1 Filesystem errors corrected
2 System should be rebooted
4 Filesystem errors left uncorrected
8 Operational error
16 Usage or syntax error

The exit code returned when all filesystems are checked using the -A option is the bitwise OR of the exit codes for each file system that is checked. In actuality, fsck is simply a front end for the various filesystem checkers (fsck.fstype) available under Linux. The filesystem-specific checker is searched for in /sbin first, then in /etc/fs and /etc, and finally in the directories listed in the PATH environment variable. Please see the filesystem-specific checker manual pages for further details.

OPTIONS

s Serialize fsck operations. Use this option if you want to check multiple file systemsin order.
-t Specifies the type of file system to be checked (for example, ext2).
-A Walk through the /etc/fstab file and try to check all file systems in one run.
-C Display completion/progress bars.
-V Produce verbose output, including all file system-specific commands that are executed.
-a Automatically repair the file system without any questions.
-r Interactively repair the file system (ask for confirmations).

Linux Runlevels

Manage runlevels using init and shutdown

Linux systems operate in different types of runlevels. A runlevel is a certain mode of operation for a system, which defines the services and processes that start for that particular runlevel.

As a Linux system boots, one process must be the first to start in order to be able to startup all other processes, daemons, and system-critical programs. This first process is called init.

init

init is the first process on a Linux system and is assigned a PID (Process Identification) of 1. The operating system kernel starts init and is responsible for starting all other services provided by the system. The services started by init are detailed in its configuration file, which is located in /etc/inittab. The inittab file, located in /etc, is the primary configuration file for the init process. The inittab file controls how your system starts up and configures the different runlevels.

Runlevel	Description
0	Halt the system.
1	Single user mode. All file systems mounted, only small set of kernel processes running. Only root can login.
2	Multi-user mode, without remote file sharing.
3	Multi-user mode with remote file sharing, processes, and daemons.
4	User definable system state.
5	Used to start X-windows on boot.
6	Shutdown and reboot.

To initiate a runlevel from the command prompt, simply enter the following command:

```
init [runlevel]
```

To bring your system to a halt, use the following command:

```
init 0
```

Most systems have their default runlevel set to 3 because they usually don't need X windows to run. If you want your system to boot and start X windows, set the default runlevel to 5 by modifying the default runlevel entry to:

```
id:5:initdefault:
```

Startup scripts

This section of the inittab file defines the location for your startup scripts for each runlevel:

```
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

The /etc/rc.d directory contains several subdirectories named after the runlevel that they represent. Each subdirectory contains scripts that are run for that particular runlevel. These scripts start a number of services and also perform a number of configuration checks, including checking the disk file systems for errors, mounting the file systems, defining swap space, cleaning out temporary files, and starting up system daemons.

Each directory contains several scripts starting with the letter "S" or the letter "K." These letters represent start and kill scripts. Depending on your current runlevel and the runlevel that you want to switch to, the system must determine which processes need to be killed and which ones need to be started.

For example, the following are some of the scripts within the /etc/rc.d/rc 3

directory:

K20nfs
K34yppasswdd
S10network
S30syslog
S60lpd
S99linuxconf

As the system switches to runlevel 3, the K20nfs will kill the nfs process, and the S10network script will start the network services.

Shutting down Linux

As with most operating systems, you need to properly shut down your Linux system to avoid damaging the system and the user data on the file systems. If your file systems are not properly unmounted, the system runs a file system check the next time the system is booted.

You may need to shut down a Linux system for several reasons, including the following:

- **General maintenance:** When you reboot a Linux system, it always performs some general maintenance tasks, such as deleting files from the temporary directories and performing checks on the machine's file systems. Restarting a system will also reset any dead or zombie processes that may be running.
- **System failure:** Often, your only recourse in the event of a system or process failure is to reboot the system. You may have to reboot for many reasons, including failure of system processes, locked devices, or runaway processes, which use up all the available CPU and RAM.
- **System upgrades:** Some system upgrades, such as a kernel update, only work if the system is rebooted. The commands that you can use to safely shut down the system are shutdown, halt, and reboot. The shutdown command is the most commonly used command for shutting down a Linux system, and there are several options that can be used for it.

For example:

```
shutdown -r now
```

The preceding option will shut down the Linux system immediately and initiate a reboot. If you don't want the system to reboot, and you'd rather have the system come to a halt, you can use the following command:

```
shutdown -h now
```

You should give your users warning before any type of scheduled shutdown in order to give them time to close their files and log out of the system. You can give the shutdown command a time parameter (in seconds) before it initiates shutdown:

```
shutdown -h 300
```

The halt command performs the same function as a shutdown -h now command, or init 0, and brings the system to a halt without rebooting. The reboot command performs the same function as a shutdown -r now command, or init 6. It brings the system down and reboots it. (Hitting the key combination, Ctrl-Alt-Delete can also start this function process).

Managing Linux Services

Stop, start, and restart services (daemons) as needed (e.g., init files)

Managing Linux services is a job function that most Linux administrators will encounter on a daily basis. Most of the services that administrators must manage are applications, such as a Web or FTP servers. However, these services can also include system processes, such as DNS, DHCP, and other important services and daemons.

Most application and service scripts are located in /etc/rc.d/init.d, or /etc/init.d. For example, the following is the startup script for the Apache Web server httpd daemon:

Most scripts utilize the arguments start, stop, restart, reload, and status.

- start: if it is not currently running, starts the process.
- stop: if it is currently running, stops the process.
- restart: stops, and then restarts the process.
- reload: restarts the process, reloading any configuration files.
- status: gives the current status of the process.

As you can see from the example httpd daemon script above, each argument will run the corresponding part of the script that will change the status of the process as directed. You must be the root user to run these scripts.

For example, if you want to restart the httpd service, use the following command:

/etc/rc.d/init.d/httpd restart

You will have to restart a process or daemon after you have changed its configuration file. The process daemon won't reload any changes until you have stopped and started the service, or performed a reload or restart. Most configuration files are located in the /etc directory. For example, the configuration file for the httpd service is /etc/httpd/conf/httpd.conf.

What is Linux Shell ?

Computer understand the language of 0's and 1's called binary language.

In early days of computing, instruction are provided using binary language, which is difficult for all of us, to read and write. So in Os there is special program called Shell. Shell accepts your instruction or commands in English (mostly) and if its a valid command, it is passed to kernel.

Shell is a user program or it's a environment provided for user interaction. Shell is an command language interpreter that executes commands read from the standard input device (keyboard) or from a file.

Shell is not part of system kernel, but uses the system kernel to execute programs, create files etc.

Several shell available with Linux including:

Shell Name	Developed by	Where	Remark
BASH (Bourne-Again SHell)	Brian Fox and Chet Ramey	Free Software Foundation	Most common shell in Linux. It's Freeware shell.
CSH (C SHell)	Bill Joy	University of California (For BSD)	The C shell's syntax and usage are very similar to the C programming language.
KSH (Korn SHell)	David Korn	AT & T Bell sLabs	
TCSH	See the man page. Type $ man tcsh		TCSH is an enhanced but completely compatible version of the Berkeley UNIX C shell (CSH).

Tip: To find all available shells in your system type following command:

$ cat /etc/shells

Note that each shell does the same job, but each understand a different command syntax and provides different built-in functions.

In MS-DOS, Shell name is COMMAND.COM which is also used for same purpose, but it's not as powerful as our Linux Shells are!

Understanding the Red Hat Linux Shell

The shell provides a way to run programs, work with the file system, compile computer code, and manage the computer. Although the shell is less intuitive than common GUIs, most Linux experts consider the shell to be much more powerful than GUIs. Because shells have been around for so long, many advanced features have been built into them.

Using the Shell in Red Hat Linux

When you type a command in a shell, you can also include other characters that change or add to how the command works. In addition to the command itself, these are some of the other items that you can type on a shell command line:

- **Options** — Most commands have one or more options you can add to change their behavior. Options typically consist of a single letter, preceded by a dash. You can also usually combine several options after a single dash. For example, the command ls -la lists the contents of the current directory. The –l asks for a detailed (long) list of information, and the -a asks that files beginning with a dot (.) also be listed. When a single option consists of a word or abbreviation, it is usually preceded by a double dash (--). For example, to use the help option on many commands, you would enter --help on the command line.
- **Arguments** — Many commands also accept arguments after any options are entered. An argument is an extra piece of information, such as a filename, that can be used by the command. For example, cat /etc/passwd prints out the contents of the /etc/passwd file. In this case, /etc/passwd is the argument.
- **Environment variables** — The shell itself stores information that may be useful to the user's shell session in what are called environment variables. Examples of environment variables include $SHELL (which identifies the shell you are using), $PS1 (which defines your shell prompt), and $MAIL (which identifies the location of your mailbox).
- **Metacharacters** — These are characters that have special meaning to the shell. Metacharacters can be used to direct the output of a command to a file (>), pipe the output to another command (|), or run a command in the background (&), to name a few.

Connecting and expanding commands

A truly powerful feature of the shell is the capability to redirect the input and output of commands to and from other commands and files.

Piping commands

The pipe (|) metacharacter connects the output from one command to the input of another command. This lets you have one command work on some data, then have the next command deal with the results. Here is an example of a command line that includes pipes:

```
$ cat /etc/password | sort | more
```

This command prints the contents of the /etc/password file and pipes the output to the sort command. The sort command takes the user names that begin each line of the /etc/password file, sorts them alphabetically, and pipes the output to the more command. The more command displays the output one page at a time, so that you can go through the output a line or a page at a time.

Sequential commands

Sometimes you may want a sequence of commands to run with one command to complete before the next command begins. You can use a semicolon (;) metacharacter to run commands in a sequence. To run a sequence of commands, type them all on the same command line and separate them with semicolons (;). For example:

```
$ date ; troff -me verylargedocument | lpr ; date
```

Background commands

Some commands can take a while to complete. Sometimes you may not want to tie up your shell waiting for a command to finish. In those cases, you can have the commands run in the background by using the ampersand (&).

Expanding commands

With command substitution, you can have the output of a command interpreted by the shell instead of by the command itself. In this way, you can have the standard output of a command become an argument for another command. The two forms of command substitution are $(command) or ‘command’. The command in this case can include options, metacharacters, and arguments. Here is an example of using command substitution:

$ vi $(find / -print | grep xyzzy)

In this command line, the command substitution is done before the vi command is run. First, the find command starts at the root directory (/) and prints out all files and directories in the entire file system. This output is piped to the grep command, which filters out all files except for those that include the string xyzzy. Finally, the vi command opens all filenames for editing (one at a time) that include xyzzy.

Expanding arithmetic expressions

There may be times when you want to pass arithmetic results to a command. There are two forms you can use to expand an arithmetic expression and pass it to the shell: $[expression] or $((expression)). Here is an example:

```
$ echo "I am $[2002 - 1957] years old."
I am 45 years old.
```

Common shell environment variables

When you start a shell (by logging in or opening a Terminal window), there is a bunch of environment variables already set. The following are some of the variables that are either set when you use a bash shell in Linux or that can be set by you to use with different features.

- **BASH** — Contains the full path name of the bash command. This is usually /bin/bash.
- **BASH_VERSION** — A number that represents the current version of the bash command.
- **ENV** — This value identifies the location of a file that contains commands used to initialize the shell. For the bash shell, this file is probably $HOME/.bashrc. See the section about creating configuration files for information on working with the .bashrc file.
- **EUID** — This is the effective user ID number of the current user. It is assigned when the shell starts.
- **HISTFILE** — The location of your history file. It is typically located at $HOME/.bash_history.
- **HISTFILESIZE** — The number of history entries that can be stored. After this number is reached, the oldest commands are discarded. The default value is 1000.
- **HISTCMD** — This returns the number of the current command in the history list.
- **HOME** — This is your home directory. It is your current working directory each time you log in or type the cd command with any options.
- **HOSTTYPE** — A value that describes the computer architecture on which the Linux system is running. For Intel-compatible PCs, the value is i386.

- **MAIL** — This is the location of your mailbox file. The file is typically your user name in the /var/spool/mail directory.
- **OLDPWD** — The directory that was the working directory before you changed to the current working directory.
- **OSTYPE** — A name identifying the current operating system. In our case, this will say Linux. (Bash can run on other operating systems as well.)
- **PATH** — The colon-separated list of directories used to find commands that you type. This value is initially set by the operating system, although most people modify it. The default value for regular users is: /usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:$HOME/bin. For the root user, the value also includes /sbin, /usr/sbin, and /usr/local/sbin.
- **PPID** — The process ID of the command that started the current shell (for example, its parent process).
- **PS1** — Sets the value of your shell prompt. There are many items that you can read into your prompt (date, time, user name, hostname, and so on). Sometimes a command requires additional prompts, which you can set with the variables PS2, PS3, and so on.
- **PWD** — This is the directory that is assigned as your current directory. This value changes each time you change directories using the cd command.
- **RANDOM** — Accessing this variable causes a random number to be generated. The number is between 0 and 99999.
- **SECONDS** — The number of seconds since the time the shell was started.
- **SHLVL** — The number of shell levels associated with the current shell session. When you log in to the shell, the SHLVL is 1. Each time you start a new bash command (by, for example, using su to become a new user, or by simply typing bash), this number is incremented.
- **TMOUT** — Can be set to a number representing the number of seconds the shell can be idle without receiving input. After the number of seconds is reached, the shell exits. This is a security feature that can help make it less likely for unattended shells to be accessed by unauthorized people. (This must be set in the login shell for it to actually cause the shell to log out the user.)

Managing background and foreground processes

If you are using Linux over a network or from a dumb terminal (a monitor that allows only text input with no GUI support), your shell may be all that you have. You may be used to a windowing environment where you have a lot of programs active at the same time so that you can switch among them as needed. This shell thing can seem pretty limited.

Though the bash shell doesn't offer you a GUI for running many programs, it does offer a way to move active programs between the background and foreground. In this way, you can have a lot of stuff running, while selectively being able to choose the one you want to deal with at the moment.

There are several ways to place an active program in the background. One mentioned earlier is to add an ampersand (&) to the end of a command line. Another way is to use the **at** command to run one or more commands in a way in which they are not connected to the shell.

To stop a running command and put it in the background, press **Ctrl**+z. After the command is stopped, you can either bring it to the foreground to run (the **fg** command) or start it running in the background (the **bg** command).

Starting background processes

If you have programs that you want to run while you continue to work in the shell, you can place the programs in the background. To place a program in the background at the time you run the program, type an ampersand (&) at the end of the command line. For example:

```
$ find /usr -print > /tmp/allusrfiles &
```

This command finds all files on your Red Hat Linux system (starting from the /usr directory), prints those file names, and puts those names in the file /tmp/allusrfiles. The ampersand (&) runs that command line in the background. To check which commands you have running in the background, use the jobs command, as follows:

```
$ jobs
[1] Stopped (tty output) vi /tmp/myfile
[2] Running find /usr -print > /tmp/allusrfiles &
[3] Running nroff -man /usr/man2/* >/tmp/man2 &
[4]- Running nroff -man /usr/man3/* >/tmp/man3 &
[5]+ Stopped nroff -man /usr/man4/* >/tmp/man4
```

The first job shows a text-editing command (vi) that was placed in the background and stopped by pressing Ctrl+z while I was editing. Job two shows the find command I just ran. Jobs three and four show nroff commands currently running in the background. Job five had been running in the shell (foreground) until I decided too many processes were running and pressed Ctrl+z to stop job five until a few processes had completed.

The plus sign (+) next to number 5 shows that this is the job that was most recently placed in the background. The minus sign (-) next to number 4 shows that it was placed in the background just before the most recent background job. Because job 1 requires terminal input, it cannot run in the background. As a result, it appears as Stopped until it is brought to the foreground again.

Using foreground and background commands

Continuing with the example shown, you can bring any of the commands on the jobs list into the foreground. For example, if you are ready to edit myfile again, you can type:

```
$ fg %1
```

As a result, the vi command opens again, with all the text as it was when you stopped the vi job.

To refer to a background job (to cancel it or bring it to the foreground), you can use a percent sign (%) followed by the job number. You can also use the following to refer to a background job:

- **% —** A percent sign alone refers to the most recent command put into the background (indicated by the plus sign). This action brings the command to the foreground.
- %string — Refers to a job where the command begins with a particular string of characters. The string must be unambiguous. (In other words, typing %vi when there are two vi commands in the background, results in an error message.)

Configuring your shell

Several configuration files support how your shell behaves. Some of these files are executed for every user and every shell. Others are specific to the particular user that creates the configuration file. Here are the files that are of interest to anyone using the bash shell in Linux:

- **/etc/profile —** This file sets up user environment information for every user. It is executed when you first log in and the shell starts. This file provides default values for your path, your prompt, the maximum file size that

you can create, and the default permissions for the files that you create. It also sets environment variables for such things as the location of your mailbox and the size of your history files.

- **/etc/bashrc —** This file is executed for every user that runs the bash shell. It is read each time a bash shell is opened. It sets the default prompt and may add one or more aliases. Values in this file can be overridden by information in each user's ~/.bashrc file.
- **~/.bash_profile —** This file is used by each user to enter information that is specific to their own use of the shell. It is executed only once, when the user logs in. By default it sets a few environment variables and executes the user's .bashrc file.
- **~/.bashrc —** This file contains the bash information that is specific to your bash shells. It is read when you log in and also each time you open a new bash shell. This is the best location to add environment variables and aliases so that your shell picks them up.
- **~/.bash_logout —** This file executes each time you log out (exit the last bash shell). By default, it simply clears your screen.

To change the /etc/profile or /etc/bashrc files, you must be the root user. Any user can change the information in the $HOME/.bash_profile, $HOME/.bashrc, and $HOME/.bash_logout files in their own home directories.

Setting your prompt

Your prompt consists of a set of characters that appear each time the shell is ready to accept a command. Exactly what that prompt contains is determined by the value in the PS1 environment variable. If your shell requires additional input, it uses the values of PS2, PS3, and PS4.

When your Red Hat Linux system is installed, your prompt is set to include the following information: your user name, your hostname, and the base name of your current working directory. That information is surrounded by brackets and followed by a dollar sign (for regular users) or a pound sign (for the root user). Here is an example of that prompt:

[student@AryanLinux student] $

You can use several special characters (indicated by adding a backslash to a variety of letters) to include different information in your prompt. These can include your terminal number, the date, and the time, as well as other pieces of information. Here are some examples:

- **\!** — Shows the current command history number. This includes all previous commands stored for your user name.
- **\#** — Shows the command number of the current command. This includes only the commands for the active shell.
- **\$** — Shows the standard user prompt ($) or root prompt (#), depending on which user you are.
- **\W** — Shows only the current working directory base name. For example, if the current working directory was /var/spool/mail, this value would simply appear as mail.
- **\[** — Precedes a sequence of nonprinting characters. This could be used to add a terminal control sequence into the prompt for such things as changing colors, adding blink effects, or making characters bold. (Your terminal determines the exact sequences available.)
- **\]** — Follows a sequence of nonprinting characters.
- \\ — Shows a backslash.
- **\d** — Displays the day, month, and number of the date. For example: Sat Jan 23.
- **\h** — Shows the hostname of the computer running the shell.
- **\n** — Causes a newline to occur.

- **\nnn —** Shows the character that relates to the octal number replacing nnn.
- **\s —** Displays the current shell name. For example, for this bash shell the value would be bash.
- **\t —** Prints the current time in hours, minutes, and seconds. For example, 10:14:39.
- **\u —** Prints your current user name.
- **\w —** Displays the full path to the current working directory.

What is Shell Script ?

Normally shells are interactive. It means shell accept command from you (via keyboard) and execute them. But if you use command one by one (sequence of 'n' number of commands) , the you can store this sequence of command to text file and tell the shell to execute this text file instead of entering the commands. This is know as ***shell script***.

Shell script defined as:

"*Shell Script is **series of command** written **in plain text file**. Shell script is just like batch file is MS-DOS but have more power than the MS-DOS batch file.*"

Why to Write Shell Script ?

- Shell script can take input from user, file and output them on screen.
- Useful to create our own commands.
- Save lots of time.
- To automate some task of day today life.
- System Administration part can be also automated.

How to write shell script

Following steps are required to write shell script:

1. Use any editor like vi or mcedit to write shell script.
2. After writing shell script set execute permission for your script as follows

syntax:

chmod permission your-script-name

Examples:

```
$ chmod +x <your-script-name>
$ chmod 755 <your-script-name>
```

Note: This will set read write execute(7) permission for owner, for group and other permission is read and execute only(5). (3) Execute your script as

syntax:

bash <your-script-name>
sh <your-script-name>
./<your-script-name>

Examples:

```
$ bash bar
$ sh bar
$ ./bar
```

NOTE: In the last syntax ./ means current directory, But only . (dot) means execute given command file in current shell without starting the new copy of shell, The syntax for . (dot) command is as follows

Syntax:

. <command-name>

Example:
```
$ . foo

$ vi first
#
# My first shell script
#
clear
echo "Knowledge is Power"
```

After saving the above script, you can run the script as follows:
```
$ ./first
```

This will not run script since we have not set execute permission for our script *first*; to do this type command
```
$ chmod 755 first
$ ./first

or
$ . first
```

Tip: For shell script file try to give file extension such as .sh, which can be easily identified by you as shell script.

Exercise:
1)Write following shell script, save it, execute it and note down it's output.
```
$ vi ginfo
#
#
# Script to print user information who currently login , current date
& time
#
clear
echo "Hello $USER"
echo "Today is \c ";date
echo "Number of user login : \c" ; who | wc -l
echo "Calendar"
cal
exit 0
```

Variables in Shell

To process our data/information, data must be kept in computers RAM memory. RAM memory is divided into small locations, and each location had unique number called memory location/address, which is used to hold our data. Programmer can give a unique name to this memory location/address called memory variable or variable (Its a named storage location that may take different values, but only one at a time).

In Linux (Shell), there are two types of variable:

1. **System variables** - Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.
2. **User defined variables (UDV)** - Created and maintained by user. This type of variable defined in lower letters.

You can see system variables by giving command like **$ set**, some of the important System variables are:

How to define User defined variables (UDV)

To define UDV use following syntax

Syntax:

<variable name>=value

'**value**' is assigned to given '**variable name**' and Value must be on right side = sign.

Example:

```
$ no=10 # this is ok
$ 10=no # Error, NOT Ok, Value must be on right side of = sign.
```

To define variable called 'vech' having value Bus

```
$ vech=Bus
```

To define variable called n having value 10

```
$ n=10
```

Rules for Naming variable name (Both UDV and System Variable)

1. Variable name must begin with Alphanumeric character or underscore character (_), followed by one or more Alphanumeric character. For e.g. Valid shell variable are as follows

   HOME
   SYSTEM_VERSION
   vech
   no

2. Don't put spaces on either side of the equal sign when assigning value to variable. For e.g. In following variable declaration there will be no error

   ```
   $ no=10
   $ no= 10
   ```

   But there will be problem for any of the following variable declaration:

   ```
   $ no =10
   $ no = 10
   ```

3. Variables are case-sensitive, just like filename in Linux. For e.g.

   ```
   $ no=10
   $ No=11
   $ NO=20
   $ nO=2
   ```

Above all are different variable name, so to print value 20 we have to use $ echo $NO and not any of the following

```
$ echo $no  # will print 10 but not 20
$ echo $No  # will print 11 but not 20
$ echo $nO  # will print 2 but not 20
```

4. You can define NULL variable as follows (NULL variable is variable which has no value at the time of definition) For e.g.

   $ vech=
   $ vech=""

   Try to print it's value by issuing following command
   ```
   $ echo $vech
   ```

   Nothing will be shown because variable has no value i.e. NULL variable.

5. Do not use **?,*** etc, to name your variable names.

How to print or access value of UDV (User defined variables)

To print or access UDV use following syntax

Syntax:
$variablename

Define variable vech and n as follows:
```
$ vech=Bus
$ n=10
```

To print contains of variable 'vech' type
```
$ echo $vech
```

It will print 'Bus', To print contains of variable 'n' type command as follows
```
$ echo $n
```

Caution: Do not try **$ echo vech**, as it will print vech instead its value 'Bus' and **$ echo n**, as it will print n instead its value '10', You must *use **$ followed by variable name***.

echo Command

Use echo command to display text or value of variable.

echo [options] [string, variables...]

Displays text or variables value on screen.

Options

-n Do not output the trailing new line.
-e Enable interpretation of the following backslash escaped characters in the strings:
\a alert (bell)
\b backspace
\c suppress trailing new line
\n new line
\r carriage return
\t horizontal tab
\\ backslash

For e.g. **$ echo -e "An apple a day keeps away \a\t\tdoctor\n"**

Shell Arithmetic

Use to perform arithmetic operations.

Syntax:

expr <op1> <math-operator> <op2>

Examples:

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo `expr 6 + 3`
```

Note:

expr 20 %3 - Remainder read as 20 mod 3 and remainder is 2.
expr 10 * 3 - Multiplication use * and not * since its wild card.

For the last statement not the following points

1. First, before **expr** keyword we used **` (back quote)** sign not the (single quote i.e. ') sign. Back quote is generally found on the key under tilde (~) on PC keyboard OR to the above of TAB key.
2. Second, expr is also end with ` i.e. **back quote**.
3. Here expr 6 + 3 is evaluated to 9, then echo command prints 9 as sum
4. Here if you use double quote or single quote, it will NOT work

$ echo "expr 6 + 3" # It will print expr 6 + 3
$ echo 'expr 6 + 3' # It will print expr 6 + 3

More about Quotes

There are three types of quotes

Quotes	Name	Meaning
"	Double Quotes	"Double Quotes" - Anything enclose in double quotes removed meaning of that characters (except \ and $).
'	Single quotes	'Single quotes' - Enclosed in single quotes remains unchanged.

`	Back quote	`Back quote` or grave quote - To execute command

Example:
$ echo "Today is date"
Can't print message with today's date.

$ echo "Today is `date`".
It will print today's date as, Today is Tue Jan ....,Can you see that the `date` statement uses back quote?

Exit Status

By default in Linux if particular command/shell script is executed, it return two type of values which is used to see whether command or shell script executed is successful or not.

1. If return *value is zero* (0), command is successful.
2. If return *value is nonzero*, command is not successful or some sort of error executing command/shell script.

This value is know as ***Exit Status***. But how to find out exit status of command or shell script?
Simple, to determine this exit Status you can use **$?** special variable of shell.

$ rm unknow1file

It will show error as follows rm: cannot remove `unkowm1file': No such file or directory
and after that if you give command

$ echo $?

it will print nonzero value to indicate error. Now give command

$ ls
$ echo $?

It will print 0 to indicate command is successful.

Exercise
Try the following commands and not down the exit status:

```
$ expr 1 + 3
$ echo $?
$ echo Welcome
$ echo $?
$ wildwest canwork?
$ echo $?
$ date
$ echo $?
$ echon $?
$ echo $?
```

The read Statement

Use to get input (data from user) from keyboard and store (data) to variable.

Syntax:

read variable1, variable2,...variableN

Following script first ask user, name and then waits to enter name from the user via keyboard. Then user enters name from keyboard (after giving name you have to press ENTER key) and entered name through keyboard is stored (assigned) to variable fname.

```
$ vi sayH
#
#Script to read your name from key-board
#
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
```

Run it as follows:

```
$ chmod 755 sayH
$ ./sayH
or

$ . sayH
```

Your first name please: ***vivek***
Hello vivek, Lets be friend!

Command Line Processing

Try the following command (assumes that the file "**grate_stories_of**" is not exist on your system)

$ ls grate_stories_of

It will print message something like - *grate_stories_of: No such file or directory*.
ls is the name of an *actual command* and shell executed this command when you type command at shell prompt. Now it creates one more question **What are commands?** What happened when you type *$ ls grate_stories_of* ?

The first word on command line is, **ls** - is name of the command to be executed.
Everything else on command line is taken *as arguments to this command*. For e.g.

$ tail +10 myf

Name of command is **tail**, and the arguments are **+10** and **myf**.

Exercise

Try to determine command and arguments from following commands

```
$ ls foo
$ cp y y.bak
$ mv y.bak y.okay
$ tail -10 myf
$ mail raj
```

```
$ sort -r -n myf
$ date
$ clear
```

$#** holds number of arguments specified on command line. And **$* or **$@** refer to all arguments passed to

Why Command Line arguments required

Telling the command/utility which option to use.

1. Informing the utility/command which file or group of files to process (reading/writing of files).
2. Let's take rm command, which is used to remove file, but which file you want to remove and how you will tell this to rm command (even rm command don't ask you name of file that you would like to remove). So what we do is we write command as follows:

$ myshell foo bar

Here **$#** (built in shell variable) will be 2 (Since foo and bar only two Arguments), Please note at a time such 9 arguments can be used from $1..$9, You can also refer all of them by using $* (which expand to `$1,$2...$9`). Note that $1..$9 i.e command line arguments to shell script is know as "*positional parameters*".

Exercise

Try to write following for commands
Shell Script Name ($0),
No. of Arguments (i.e. $#),
And actual argument (i.e. $1,$2 etc)

Following script is used to print command ling argument and will show you how to access them:

```
$ vi demo
```

```
#!/bin/sh
#
# Script that demos, command line args
#
echo "Total number of command line argument are $#"
echo "$0 is script name"
echo "$1 is first argument"
echo "$2 is second argument"
echo "All of them are :- $* or $@"
```

Run it & test it as follows:

$ ./demo Hello World
If test successful, copy script to your own bin directory (Install script for private use)

$ cp demo ~/bin
Check whether it is working or not (?)

$ demo

$ demo Hello World

NOTE: After this, for any script you have to used above command, in sequence, I am not going to show you all of the above command(s) for rest of Tutorial.

Also note that you ***can't assigne the new value to command line arguments i.e positional parameters***. So following all statements in shell script are invalid:
$1 = 5
$2 = "My Name"

Redirection of Standard output/input i.e. Input - Output redirection

Mostly all commands give output on screen or take input from keyboard, but in Linux (and in other OS's also) it's possible to send output to file or to read input from file. For e.g.

$ ls command gives output to screen; to send output to file of ls command give command
$ ls > filename

It means put output of ls command to filename.

There are three main redirection symbols **>,>>,<**

1. > Redirector Symbol

   Syntax:
   Linux-command > filename

   To output Linux-commands result (output of command or shell script) to file. Note that if file already exist, it will be overwritten else new file is created. For e.g. To send output of ls command give

   $ ls > myfiles

   Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

2. >> Redirector Symbol

   Syntax:

   Linux-command >> filename

   To output Linux-commands result (output of command or shell script) to END of file. Note that if file exist , it will be opened and new information/data will be written to END of file, without losing previous information/data, And if file is not exist, then new file is created. For e.g. To send output of date command to already exist file give command

   $ date >> myfiles

   < Redirector Symbol

Syntax:

Linux-command < filename

To take input to Linux-command from file instead of key-board. For e.g. To take input for cat command give

$ cat < myfiles
$cat > sname
vivek
ashish
zebra
babu

Press CTRL + D to save.

Now issue following command.

$ sort < sname > sorted_names
$ cat sorted_names
ashish
babu
vivek
zebra

In above example sort (**$ sort < sname > sorted_names**) command takes input from sname file and output of sort command (i.e. sorted names) is redirected to sorted_names file.

Try one more example to clear your idea:

$ tr "[a-z]" "[A-Z]" < sname > cap_names
$ cat cap_names
VIVEK
ASHISH
ZEBRA
BABU

tr command is used to translate all lower case characters to upper-case letters. It take input from sname file, and tr's output is redirected to cap_names file.

Pipes

A pipe is a way to connect the output of one program to the input of another program without anytemporary file.

Pipe Defined as:

"A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands Multiple commands) from same command line."

Syntax:

command1 | command2

Filter

If a Linux command accepts its input from the standard input and produces its output on standard output is know as a filter. A filter performs some kind of process on the input and gives output. For e.g.. Suppose you have file called 'hotel.txt' with 100 lines data, And from 'hotel.txt' you would like to print contains from line number 20 to line number 30 and store this result to file called 'hlist' then give command:

$ tail +20 < hotel.txt | head -n30 >hlist

Here **head** command is filter which takes its input from tail command (tail command start selecting from line number 20 of given file i.e. hotel.txt) and passes this lines as input to head, whose output is redirected to 'hlist' file.

Consider one more following example

$ sort < sname | uniq > u_sname

Here uniq is filter which takes its input from sort command and passes this lines as input to uniq; Then uniqs output is redirected to "u_sname" file.

Introduction

Making decision is important part in ONCE life as well as in computers logical driven program. In fact logic is not LOGIC until you use decision making. This chapter introduces to the bash's structured language constructs such as:

- Decision making
- Loops

Is there any difference making decision in Real life and with Computers? Well real life decisions are quite complicated to all of us and computers even don't have that much power to understand our real life decisions. What computer know is 0 (zero) and 1 that is Yes or No.

if condition

if condition which is used for decision making in shell script, If given condition is true then command1 is executed.

Syntax:

```
if <condition>
then
        command1 if condition is true or if exit status
        of condition is 0 (zero)
        ...
        ...
fi
```

Condition is defined as:

"*Condition is nothing but comparison between two values.*"

For compression you can use **test** or **[expr]** statements or even exist status can be also used.
Expreession is defined as:

"*An expression is nothing but combination of values, relational operator (such as >,<, <> etc) and mathematical operators (such as +, -, / etc).*"

Following are all examples of expression:

5 > 2
3 + 6
3 * 65
a < b
c > 5
c > 5 + 30 –1

Type following commands (assumes you have file called **foo**)
$ cat foo
$ echo $?

The cat command return zero(0) i.e. exit status, on successful, this can be used, in if condition as follows,

Write shell script as

```
$ cat > showfile
#!/bin/sh
#
#Script to print file
#
if cat $1
then
echo -e "\n\nFile $1, found and successfully echoed"
fi
```

Run above script as:
$ chmod 755 showfile
$./showfile foo

Shell script name is showfile ($0) and foo is argument (which is $1). Then shell compare it as follows:

if cat $1 which is expanded to if cat foo.

Detailed explanation

if cat command finds foo file and if its successfully shown on screen, it means our cat command is successful and its exist status is 0 (indicates success), So our if condition is also true and hence statement echo -e "\n\nFile $1, found and successfully echoed" is proceed by shell. Now if cat command is not successful then it returns non-zero value (indicates some sort of failure) and this statement echo –e "\n\nFile $1, found and successfully echoed" is skipped by our shell.

Exercise
Write shell script as follows:

```
cat > trmif
#
# Script to test rm command and exist status
```

```
#
if rm $1
then
echo "$1 file deleted"
fi
```

Press Ctrl + d to save

test command or [expr]

test command or [expr] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.

Syntax:

test expression OR [expression]

Example:

Following script determine whether given argument number is positive.

```
$ cat > ispositive
#!/bin/sh
#
# Script to see whether argument is positive
#
if test $1 -gt 0
then
echo "$1 number is positive"
fi
```

Run it as follows

$ chmod 755 ispostive

$ ispostive 5

5 number is positive

$ispostive -45

Nothing is printed

$ispostive

./ispostive: test: -gt: unary operator expected

Detailed explanation

test or [expr] works with

1.Integer (Number without decimal point)
2.File types
3.Character strings

For Mathematics, use following operator in Shell Script

Mathematical Operator in Shell Script	Meaning	Normal Arithmetical/ Mathematical Statements	But in Shell	
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if [5 -ge 6]

NOTE: == is equal, != is not equal.

For string Comparisons use

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Shell also test for file and directory types

Test	Meaning
-s	file Non empty file
-f	file Is File exist or normal file and not a directory
-d	dir Is Directory exist and not a file
-w	file Is writeable file
-r	file Is read-only file
-x	file Is file is executable

Logical Operators

Logical operators are used to combine two or more condition at a time

Operator	Meaning
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

if...else...fi

If given condition is true then command1 is executed otherwise command2 is executed.

Syntax:

```
if <condition>
then
     condition is zero (true - 0)
     execute all commands up to else statement
else
     if condition is not true then
     execute all commands up to fi
fi
```

For e.g. Write Script as follows:

```
$ vi isnump_n
#!/bin/sh
#
# Script to see whether argument is positive or negative
#
if [ $# -eq 0 ]
then
echo "$0 : You must give/supply one integers"
exit 1
fi
if test $1 -gt 0
then
echo "$1 number is positive"
else
echo "$1 number is negative"
fi
```

Try it as follows:

$ chmod 755 isnump_n
$ isnump_n 5
5 number is positive
$ isnump_n -45
-45 number is negative
$ isnump_n
./ispos_n : You must give/supply one integers
$ isnump_n 0
0 number is negative

Nested if-else-fi

You can write the entire if-else construct within either the body of the if statement of the body of an else statement. This is called the nesting of ifs.

```
$ vi nestedif.sh
osch=0
echo "1. Unix (Sun Os)"
echo "2. Linux (Red Hat)"
echo -n "Select your os choice [1 or 2]? "
```

```
read osch
if [ $osch -eq 1 ] ; then
echo "You Pick up Unix (Sun Os)"
else #### nested if i.e. if within if ######
if [ $osch -eq 2 ] ; then
echo "You Pick up Linux (Red Hat)"
else
echo "What you don't like Unix/Linux OS."
fi
fi
```

Run the above shell script as follows:
$ chmod +x nestedif.sh
$ ./nestedif.sh
1. Unix (Sun Os)
2. Linux (Red Hat)
*Select you os choice [1 or 2]? **1***
You Pick up Unix (Sun Os)
$ ./nestedif.sh
1. Unix (Sun Os)
2. Linux (Red Hat)
*Select you os choice [1 or 2]? **2***
You Pick up Linux (Red Hat)
$ ./nestedif.sh
1. Unix (Sun Os)
2. Linux (Red Hat)
*Select you os choice [1 or 2]? **3***
What you don't like Unix/Linux OS.

Note that Second *if-else* constuct is nested in the first *else* statement. If the condition in the first *if* statement is false the the condition in the second *if* statement is checked. If it is false as well the final *else* statement is executed.

You can use the nested *if*s as follows also:

Syntax:

```
     if condition
     then
          if condition
          then
               .....
               ..
               do this
          else
               ....
               ..
               do this
          fi
     else
          ...
          .....
          do this
```

```
fi
```

Multilevel if-then-else

Syntax:

```
if condition
then
     condition is zero (true - 0)
     execute all commands up to elif statement
elif condition1
then
     condition1 is zero (true - 0)
     execute all commands up to elif statement
elif condition2
then
     condition2 is zero (true - 0)
     execute all commands up to elif statement
else
     None of the above condtion,condtion1,condtion2 are true (i.e.
     all of the above nonzero or false)
     execute all commands up to fi
fi
```

For multilevel if-then-else statement try the following script:

```
$ cat > elf
#
#!/bin/sh
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
echo "$1 is positive"
elif [ $1 -lt 0 ]
then
echo "$1 is negative"
elif [ $1 -eq 0 ]
then
echo "$1 is zero"
else
echo "Opps! $1 is not number, give number"
fi
```

Try above script as follows:

$ chmod 755 elf
$ ./elf 1
$ ./elf -2
$ ./elf 0
$ ./elf a

Here o/p for last sample run:

./elf: [: -gt: unary operator expected
./elf: [: -lt: unary operator expected
./elf: [: -eq: unary operator expected
Opps! a is not number, give number

The case Statement

The case statement is good alternative to Multilevel if-then-else-fi statement. It enable you to match several values against one variable. Its easier to read and write.

Syntax:

```
case $variable-name in
    pattern1) command
        ...
        ..
        command;;
    pattern2) command
        ...
        ..
        command;;
    patternN) command
        ...
        ..
        command;;
    *) command
        ...
        ..
        command;;
    esac
```

The *$variable-name* is compared against the patterns until a match is found. The shell then executes all the statements up to the two semicolons that are next to each other. The default is *) and its executed if no match is found. For e.g. write script as follows:

```
$ cat > car
#
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg
if [ -z $1 ]
then
rental="*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
rental=$1
fi
case $rental in
"car") echo "For $rental Rs.20 per k/m";;
"van") echo "For $rental Rs.10 per k/m";;
"jeep") echo "For $rental Rs.5 per k/m";;
"bicycle") echo "For $rental 20 paisa per k/m";;
*) echo "Sorry, I can not gat a $rental for you";;
esac
```

Save it by pressing CTRL+D and run it as follows:
$ chmod +x car
$ car van
$ car car
$ car Maruti-800

Loops in Shell Scripts

Loop defined as:

"*Computer can repeat particular instruction again and again, until particular condition satisfies. A group of instruction that is executed repeatedly is called a loop.*"

Bash supports:

- for loop
- while loop

Note that in each and every loop,

1. First, the variable used in loop condition must be initialized, then execution of the loop begins.
2. A test (condition) is made at the beginning of each iteration.
3. The body of loop ends with a statement that modifies the value of the test (condition) variable.

for Loop

Syntax:

```
for { variable name } in { list }
do
        execute one for each item in the list until the list is not finished (And repeat all
        statement between do and done)
done
```

Before try to understand above syntax try the following script:

```
$ cat > testfor
for i in 1 2 3 4 5
do
echo "Welcome $i times"
done
```

Run it above script as follows:
$ chmod +x testfor
$ ./testfor

The for loop first creates i variable and assigned a number to i from the list of number from 1 to 5, The shell execute echo statement for each assignment of i. (This is usually know as iteration) This process will continue until all the items in the list were not finished, because of this it will repeat 5 echo statements. To make you idea more clear try following script:

```
$ cat > mtable
#!/bin/sh
#
#Script to test for loop
#
#
if [ $# -eq 0 ]
```

```
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
echo "Use to print multiplication table for given number"
exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

Save above script and run it as:
$ chmod 755 mtable
$ ./mtable 7
$ ./mtable

For first run, above script print multiplication table of given number where i = 1,2 ... 10 is multiply by given n (here command line argument 7) in order to produce multiplication table as
7 * 1 = 7
7 * 2 = 14
...
..
7 * 10 = 70

And for second test run, it will print message -
Error - Number missing form command line argument
Syntax : ./mtable number
Use to print multiplication table for given number

Note that to terminate our script we used 'exit 1' command which takes 1 as argument (1 indicates error and therefore script is terminated)

Even you can use following syntax:

Syntax:

```
for (( expr1; expr2; expr3 ))
do
     .....
     ...
     repeat all statements between do and
     done until expr2 is TRUE
done
```

In above syntax BEFORE the first iteration, ***expr1*** is evaluated. This is usually used to initialize variables for the loop. All the statements between do and done is executed repeatedly UNTIL the value of ***expr2*** is TRUE. AFTER each iteration of the loop, ***expr3*** is evaluated. This is usually use to increment a loop counter.

$ cat > for2
for ((i = 0 ; i <= 5; i++))
do

```
echo "Welcome $i times"
done
```

Run the above script as follows:
$ chmod +x for2
$ ./for2
Welcome 0 times
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
LSST v1.05r3 > Chapter 3 > for Loop
Welcome 5 times

Nesting of for Loop

As you see the if statement can nested, similarly loop statement can be nested. You can nest the for loop. To understand the nesting of for loop see the following shell script.

```
$ vi nestedfor.sh
for (( i = 1; i <= 5; i++ )) ### Outer for loop ###
do
for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###
do
echo -n "$i "
done
echo "" #### print the new line ###
done
```

Run the above script as follows:
$ chmod +x nestedfor.sh
$ ./nestefor.sh
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5

Following script is quite intresting, it prints the chess board on screen.

```
$ vi chessboard
for (( i = 1; i <= 9; i++ )) ### Outer for loop ###
do
for (( j = 1 ; j <= 9; j++ )) ### Inner for loop ###
do
tot=`expr $i + $j`
tmp=`expr $tot % 2`
if [ $tmp -eq 0 ]; then
echo -e -n "\033[47m "
else
echo -e -n "\033[40m "
fi
```

```
done
echo -e -n "\033[40m" #### set back background colour to black
echo "" #### print the new line ###
done
```

Run the above script as follows:

$ chmod +x chessboard
$ ./chessboard

while loop

Syntax:

while [condition]
do
command1
command2
command3
..
....
done

Loop is executed as long as given condition is true. For e.g.. Above for loop program (shown in last section of for loop) can be written using while loop as:

```
$cat > nt1
#!/bin/sh
#
#Script to test while statement
#
#
if [ $# -eq 0 ]
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
echo " Use to print multiplication table for given number"
exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
do
echo "$n * $i = `expr $i \* $n`"
i=`expr $i + 1`
done
```

Save it and try as

$ chmod 755 nt1
$./nt1 7

How to de-bug the shell script?

While programming shell sometimes you need to find the errors (bugs) in shell script and correct the errors (remove errors - debug). For this purpose you can use -v and -x option with sh or bash command to debug the shell script. General syntax is as follows:

Syntax:

sh option { shell-script-name }

OR

bash option { shell-script-name }

Option can be

-v Print shell input lines as they are read.

-x After expanding each simple-command, bash displays the expanded value of PS4 system variable, followed by the command and its expanded arguments.

Example:

```
$ cat > dsh1.sh
#
# Script to show debug of shell
#
tot=`expr $1 + $2`
echo $tot
```

Press ctrl + d to save, and run it as

$ chmod 755 dsh1.sh

$ ./dsh1.sh 4 5

9

$ sh -x dsh1.sh 4 5

```
#
# Script to show debug of shell
#
tot=`expr $1 + $2`
expr $1 + $2
++ expr 4 + 5
+ tot=9
echo $tot
+ echo 9
9
```

See the above output, -x shows the exact values of variables (or statements are shown on screen with values).

$ sh -v dsh1.sh 4 5

Use -v option to debug complex shell script.

/dev/null - Use to send unwanted output of program

This is special Linux file, which is used to send any unwanted output from program/command.

Syntax:

command > /dev/null

Example:
$ ls > /dev/null

Output of above command is not shown on screen its send to this special file. The /dev directory contains other device files. The files in this directory mostly represent peripheral devices such disks like floppy disk, sound card, line printers etc. See the file system tutorial for more information on Linux disk, partition and file system.

Future Point:
Run the following two commands
$ ls > /dev/null
$ rm > /dev/null

Local and Global Shell variable (export command)

Normally all our variables are local. Local variable can be used in same shell, if you load another copy of shell (by typing the **/bin/bash** at the $ prompt) then new shell ignored all old shell's variable. For e.g. Consider following example
$ vech=Bus
$ echo $vech
Bus
$ /bin/bash
$ echo $vech

NOTE:-Empty line printed
$ vech=Car
$ echo $vech
Car
$ exit
$ echo $vech
Bus

Global shell defined as:
"*You can copy old shell's variable to new shell (i.e. first shells variable to seconds shell), such variable is know as Global Shell variable.*"

To set global varible you have to use export command.

Syntax:

export variable1, variable2,.....variableN

Examples:
$ vech=Bus
$ echo $vech
Bus

$ export vech
$ /bin/bash
$ echo $vech
Bus
$ exit
$ echo $vech
Bus

Conditional execution i.e. && and ||

The control operators are && (read as AND) and || (read as OR). The syntax for AND list is as follows

Syntax:

command1 && command2

command2 is executed if, and only if, command1 returns an exit status of zero.

The syntax for OR list as follows

Syntax:

command1 || command2

command2 is executed if and only if command1 returns a non-zero exit status.

You can use both as follows
Syntax:

command1 && comamnd2 if exist status is zero || command3 if exit status is non-zero

if command1 is executed successfully then shell will run command2 and if command1 is not successful then command3 is executed.

Example:
$ rm myf && echo "File is removed successfully" || echo "File is not removed"

If file (myf) is removed successful (exist status is zero) then "*echo File is removed successfully*" statement is executed, otherwise "*echo File is not removed*" statement is executed (since exist status is non-zero)

The shift Command

The shift command moves the current values stored in the positional parameters (command line args) to the left one position. For example, if the values of the current positional parameters are: $1 = -f $2 = foo $3 = bar and you executed the shift command the resulting positional parameters would be as follows:

$1 = foo $2 = bar

For e.g. Write the following shell script to clear you idea:

```
$ vi shiftdemo.sh
echo "Current command line args are: \$1=$1, \$2=$2, \$3=$3"
shift
echo "After shift command the args are: \$1=$1, \$2=$2, \$3=$3"
```

Excute above script as follows:
$ chmod +x shiftdemo.sh
$ ./shiftdemo -f foo bar

Current command line args are: $1=-f, $2=foo, $3=bar
After shift command the args are: $1=foo, $2=bar, $3=

You can also move the positional parameters over more than one place by specifying a number with the shift command. The following command would shift the positional parameters two places: shift 2

But where to use shift command?

You can use shift command to parse the command line (args) option. For example consider the following simple shell script:

```
$ vi convert
while [ "$1" ]
do
if [ "$1" = "-b" ]; then
ob="$2"
case $ob in
16) basesystem="Hex";;
8) basesystem="Oct";;
2) basesystem="bin";;
*) basesystem="Unknown";;
esac
shift 2
elif [ "$1" = "-n" ]
then
num="$2"
shift 2
else
echo "Program $0 does not recognize option $1"
exit 1
fi
done
output=`echo "obase=$ob;ibase=10; $num;" | bc`
echo "$num Decimal number = $output in $basesystem number
system(base=$ob)"
```

Save and run the above shell script as follows:
$ chmod +x convert
$ ./convert -b 16 -n 500
500 Decimal number = 1F4 in Hex number system(base=16)
$ ./convert -b 8 -n 500
500 Decimal number = 764 in Oct number system(base=8)
$ ./convert -b 2 -n 500
500 Decimal number = 111110100 in bin number system(base=2)
$ ./convert -b 2 -v 500
Program ./convert does not recognize option -v
$ ./convert -t 2 -v 500
Program ./convert does not recognize option -t
$ **./convert -b 4 -n 500**
500 Decimal number = 13310 in Unknown number system(base=4)
$ ./convert -n 500 -b 16
500 Decimal number = 1F4 in Hex number system(base=16)

Above script is run in variety of ways. First three sample run converts the number 500 (-n 500) to respectively 1F4 (hexadecimal number i.e. -b 16), 764 (octal number i.e. -b 16) , 111110100 (binary number i.e. -b 16). It use **-n** and **-b** as command line option which means:

-b {base-system i.e. 16,8,2 to which *-n number* to convert}

n {Number to convert to *-b base-system*}

Last sample run shows that command line options can given different ways i.e. you can use it as follows:
$ ./convert -n 500 -b 16
Instead of
$ ./convert -b 16 -n 500

Prepering for Quick Tour of essential utilities

For this part of tutorial create *sname* and *smark* data files as follows (Using text editor of your choice)

Note Each data block is separated from the other by TAB character i.e. while creating the file if you type 11 then press "tab" key, and then write Vivek (as shown in following files):

sname

Sr.No	Name
11	Vivek
12	Renuka
13	Prakash
14	Ashish
15	Rani

smark

Sr.No	Mark
11	67
12	55
13	96
14	36
15	67

Selecting portion of a file using cut utility

Suppose from *sname* file you wish to print name of student on-screen, then from shell (Your command prompt i.e. $) issue command as follows:

$cut -f2 sname
Vivek
Renuka
Prakash
Ashish
Ran
i

cut utility cuts out selected data from *sname* file. To select Sr.no. field from sname give command as follows:

$cut -f1 sname
11
12
13
14
15

Command	Explanation
cut	Name of cut utility
-f1	Using (-f) option, you are specifying the extraction field number. (In this example its 1 i.e. first field)
sname	File which is used by cut utility and which is use as input for cut utility.

You can redirect output of cut utility as follows
$cut -f2 sname > /tmp/sn.tmp.$$
$cut -f2 smark > /tmp/sm.tmp.$$
$cat /tmp/sn.tmp.$$
Vivek
Renuka
Prakash
Ashish
Rani
$cat /tmp/sm.tmp.$$
67
55
96
36
67
General Syntax of cut utility:
Syntax:
cut -f{field number} {file-name}
Use of Cut utility:
Selecting portion of a file.

Putting lines together using **paste** utility

Now enter following command at shell prompt

$ paste sname smark
11 Vivek 11 67
12 Renuka 12 55
13 Prakash 13 96
14 Ashish 14 36
15 Rani 15 67

Paste utility join *textual information together*. To clear your idea try following command at shell prompt:

$ paste /tmp/sn.tmp.$$ /tmp/sm.tmp.$$
Vivek 67

Renuka 55
Prakash 96
Ashish 36
Rani 67

Paste utility is useful to put textual information together located in various files.
General Syntax of paste utility:

Syntax:

paste {file1} {file2}

Use of paste utility:

Putting lines together. Can you note down basic difference between cut and paste utility?

The join utility

Now enter following command at shell prompt:

$join sname smark
11 Vivek 67
12 Renuka 55
13 Prakash 96
14 Ashish 36
15 Rani 67

Here students names are matched with their appropriate marks. How ? join utility uses the Sr.No. field to join to files. Notice that Sr.No. is the first field in both sname and smark file.

General Syntax of join utility:

Syntax:

join {file1} {file2}

Use of join utility:

The join utility joins, lines from separate files.
Note t hat join will only work, if there is *common field in both file and if values are identical to each other*.

Translateing range of characters using tr utility

Type the following command at shell prompt:

$ tr "h2" "3x" < sname
11 Vivek
1x Renuka
13 Prakas3
14 As3is3
15 Rani

You can clearly see that each occurrence of character 'h' is replace with '3' and '2' with 'x'. tr utility translate specific characters into other specific characters or range of characters into other ranges.
h -> 3
2 -> x

Consider following example: (after executing command type text in lower case)
$ tr "[a-z]" "[A-Z]"
hi i am Vivek
HI I AM VIVEK
what a magic
WHAT A MAGIC
{Press CTRL + C to terminate.}

Here tr translate range of characters (i.e. small a to z) into other (i.e. to Capital A to Z) ranges. General Syntax & use of tr utility:

Syntax:

tr {pattern-1} {pattern-2}

Use of tr utility:

To translate range of characters into other range of characters. After typing following paragraph, I came to know my mistake that entire paragraph must be in lowercase characters, how to correct this mistake? (Hint - Use tr utility)

$ cat > lcommunity.txt
THIS IS SAMPLE PARAGRAPH
WRITTEN FOR LINUX COMMUNITY,
BY VIVEK G GITE (WHO ELSE?)
OKAY THAT IS OLD STORY.

Data manipulation using awk utility

Before learning more about awk create data file using any text editor or simply vi:
inventory
egg order 4
cacke good 10
cheese okay 4
pen good 12
floppy good 5

After crating file issue command
$ awk '/good/ { print $3 }' inventory
10
12
5

awk utility, select each record from file containing the word "good" and performs the action of printing the third field (Quantity of available goods.). Now try the following and note down its output.

$ awk '/good/ { print $1 " " $3 }' inventory

General Syntax of awk utility:

Syntax:

awk 'pattern action' {file-name}

For **$ awk '/good/ { print $3 }' inventory** example

awk - Revisited

awk utility is powerful data manipulation/scripting programming language (In fact based on the C programming Language). Use awk to handle complex task such as calculation, database handling, report creation etc.

General Syntax of awk:

Syntax:

awk -f {awk program file} filename

awk Program contains are something as follows:

```
Pattern {
        action 1
        action 2
        action N
}
```

awk reads the input from given file (or from stdin also) one line at a time, then each line is compared with pattern. If pattern is match for each line then given action is taken. Pattern can be regular expressions. Following is the summery of common awk metacharacters:

Metacharacter	**Meaning**
. (Dot)	Match any character
*	Match zero or more character
^	Match beginning of line
$	Match end of line
\	Escape character following
[ ]	List
{ }	Match range of instance
+	Match one more preceding
?	Match zero or one preceding
|	Separate choices to match

Getting Starting with awk

Consider following text database file

Sr.No	**Product**	**Qty**	**UnitPrice**
1	Pen	5	20.00
2	Rubber	10	2.00
3	Pencil	3	3.50
4	Cock	2	45.50

In above file fields are Sr.No,Product,Qty,Unit Price. Field is the smallest element of any record. Each fields has its own attributes. For e.g. Take Qty. field. Qty. fields attribute is its numerical (Can contain only numerical data). Collection of fields is know as record. So 1. Pen 5 20.00 ----> Is a Record. Collection of record is know as *database file*. In above text database file each field is separated using space (or tab character) and record is separated using new-line character (i.e. each record is finished at the end of line). In the awk, fields are access using special variable. For e.g. In above database $1, $2, $3, $4 respectively represents Sr.No, Product, Qty, Unit Price fields. (Don't confuse $1,$2 etc with command line arguments of shell script).

For this part of tutorial create text datafile **inven** (Shown as above). Now enter following simple awk program/command at shell prompt:

$ awk '{ print $1 $2 "--> Rs." $3 * $4 }' inven
1.Pen--> Rs.100
2.Pencil--> Rs.20
3.Rubber--> Rs.10.5
4.Cock--> Rs.91

$ awk '{ print $2 }' inven
Pen
Pencil
Rubber
Cock

awk prints second field from file. Same way if you want to print second and fourth field from file then give following command:

$awk '{ print $2 $4}' inven
Pen20.00
Pencil2.00
Rubber3.50
Cock45.50

$0 is special variable of awk , which print entire record, you can verify this by issuing following awk command:

$ awk '{ print $0 }' inven
1. Pen 5 20.00
2. Pencil 10 2.00
3. Rubber 3 3.50
4. Cock 2 45.50

You can also create awk command (program) file as follows:
```
$ cat > prn_pen
/Pen/ { print $3 }
```

And then you can execute or run above "prn_pen" awk command file as follows
$ awk -f prn_pen inven
5
10

In above awk program */Pen/* is the search pattern, if this pattern is found on line (or record) then print the third field of record. *{ print $3 }* is called Action. On shell prompt , *$ awk -f prn_pen inven* , **-f** option instruct awk, to read its command from given file, *inven* is the name of database file which is taken as input for awk.

Now create following awk program as follows:
```
$cat > comp_inv
3 > 5 { print $0 }
```

Run it as follows:
$ awk -f comp_inv inven
2. Pencil 10 2.00

Here third field of database is compared with 5, this the pattern. If this pattern found on any line database, then entire record is printed.

Predefined variable of awk

Our next example talks more about predefined variable of awk. Create awk file as follows:

```
$cat > def_var
{
print "Printing Rec. #" NR "(" $0 "),And # of field for this record is
" NF
}
```

Run it as follows.
$awk -f def_var inven
Printing Rec. #1(1. Pen 5 20.00),And # of field for this record is 4
Printing Rec. #2(2. Pencil 10 2.00),And # of field for this record is 4
Printing Rec. #3(3. Rubber 3 3.50),And # of field for this record is 4
Printing Rec. #4(4. Cock 2 45.50),And # of field for this record is 4

NR and *NF* are predefined variables of awk which means Number of input Record, Number of Fields in input record respectively. In above example NR is changed as our input record changes, and NF is constant as there are only 4 field per record. Following table shows list of such built in awk variables.

awk Variable	Meaning
FILENAME	Name of current input file
RS	Input record separator character (Default is new line)
OFS	Output field separator string (Blank is default)
ORS	Output record separator string (Default is new line)
NF	Number of input record
NR	Number of fields in input record
OFMT	Output format of number
FS	Field separator character (Blank & tab is default)

Doing arithmetic with awk

You can easily, do the arithmetic with awk as follows

```
$ cat > math
{
print $1 " + " $2 " = " $1 + $2
print $1 " - " $2 " = " $1 - $2
print $1 " / " $2 " = " $1 / $2
print $1 " x " $2 " = " $1 * $2
print $1 " mod " $2 " = " $1 % $2
}
```

Run the awk program as follows:
$ awk -f math
20 3
20 + 3 = 23
20 - 3 = 17
20 / 3 = 6.66667
20 x 3 = 60
20 mod 3 = 2

(Press CTRL + D to terminate)

In above program print $1 " + " $2 " = " $1 + $2, statement is used for addition purpose. Here $1 + $2, means add (+) first field with second field. Same way you can do - (subtraction), * (Multiplication), / (Division), % (modular use to find remainder of division operation).

User Defined variables in awk

You can also define your own variable in awk program, as follows:

```
$ cat > math1
{
no1 = $1
no2 = $2
ans = $1 + $2
print no1 " + " no2 " = " ans
}
```

Run the program as follows
$ awk -f math1
1 5
1 + 5 = 6

In the above program, no1, no2, ans all are user defined variables. Value of first and second field are assigned to no1, no2 variable respectively and the addition to ans variable. Value of variable can be printed using print statement as, print no1 " + " no2 " = " ans. Note that print statement prints whatever enclosed in double quotes (" text ") as it is. If string is not enclosed in double quotes its treated as variable. Also above two program takes input from stdin (Keyboard) instead of file.

Now try the following awk program and note down its output.

```
$ cat > bill
{
total = $3 * $4
recno = $1
item = $2
print recno item " Rs." total
}
```

Run it as
$ awk -f bill inven
1.Pen Rs.100
2.Pencil Rs.20

3.Rubber Rs.10.5
4.Cock Rs.91

Here we are printing the total price of each product (By multiplying third field with fourth field). Following program prints total price of each product as well as the Grand total of all product in the bracket.

```
$ cat > bill1
{
total = $3 * $4
recno = $1
item = $2
gtotal = gtotal + total
print recno item " Rs." total " [Total Rs." gtotal "] "
}
```

Run the above awk program as follows:
$ awk -f bill1 inven
1.Pen Rs.100 [Total Rs.100]
2.Pencil Rs.20 [Total Rs.120]
3.Rubber Rs.10.5 [Total Rs.130.5]
4.Cock Rs.91 [Total Rs.221.5]

In this program, gtotal variable holds the grand total. It adds the total of each product as gtotal = gtotal + total. Finally this total is printed with each record in the bracket. But their is one problem with our script, Grand total mostly printed at the end of all record. To solve this problem we have to use special BEGIN and END Patterns of awk. First take the example,

```
$ cat > bill2
BEGIN {
print "--------------------------"
print "Bill for the 4-March-2001. "
print "By Vivek G Gite. "
print "--------------------------"
}
{
total = $3 * $4
recno = $1
item = $2
gtotal += total
print recno item " Rs." total
}
END {
print "--------------------------"
print "Total Rs." gtotal
print "=========================="
}
```

Run it as
$awk -f bill2 inven

Bill for the 4-March-2001.
By Vivek G Gite.

```
--------------------------
1.Pen Rs.100
2.Pencil Rs.20
3.Rubber Rs.10.5
4.Cock Rs.91
--------------------------
Total Rs.221.5
==============
```

instruct awk, that perform BEGIN actions before the first line (Record) has been read from database file. Use BEGIN pattern to set value of variables, to print heading for report etc. General syntax of BEGIN is as follows

Syntax:

```
BEGIN {
        action 1
        action 2
        action N
      }
```

END instruct awk, that perform END actions after reading all lines (RECORD) from the database file. General syntax of END is as follows:

```
END {
      action 1
      action 2
      action N
      }
```

Use of printf statement

Next example shows the use of special printf statement

```
$ cat > bill3
BEGIN {
printf "Bill for the 4-March-2001.\n"
printf "By Vivek G Gite.\n"
printf "--------------------------\n"
}
{
total = $3 * $4
recno = $1
item = $2
gtotal += total
printf "%d %s Rs.%f\n", recno, item, total
#printf "%2d %-10s Rs.%7.2f\n", recno, item, total
}
END {
printf "--------------------------\n"
printf "Total Rs. %f\n" ,gtotal
#printf "\tTotal Rs. %7.2f\n" ,gtotal
printf "==========================\n"
}
```

Run it as follows:

$ awk -f bill3 inven
Bill for the 4-March-2001.
By Vivek G Gite.
---
1 Pen Rs.100.000000
2 Pencil Rs.20.000000
3 Rubber Rs.10.500000
4 Cock Rs.91.000000

Total Rs. 221.500000
==========

In above example printf statement is used to print formatted output of the variables or text.

General syntax of printf as follows:

Syntax:

printf "format" ,var1, var2, var N

If you just want to print any text using printf as follows

printf "Hello"
printf "Hello World\n"

In last example **\n** is used to print new line. Its Part of escape sequence following may be also used:

\t for tab
\a Alert or bell
\" Print double quote etc

For e.g. **printf "\nAn apple a day, keeps away\t\t\tDoctor\n\a\a"**

It will print text on new line as : *An apple a day, keeps away Doctor*

To run above example simply create any awk program file as follows

```
$ cat > p_demo
BEGIN {
n = 10
printf "%d", n
printf "\nAn apple a day, keeps away\t\t\tDoctor\n\a\a"
}
```

Run it as
$ awk -f p_demo
10
An apple a day, keeps away Doctor

Use of Format Specification Code

```
$ cat > bill4
BEGIN {
printf "Bill for the 4-March-2001.\n"
printf "By Vivek G Gite.\n"
printf "--------------------------\n"
```

```
}
{
total = $3 * $4
recno = $1
item = $2
gtotal += total
printf "%2d %-10s Rs.%7.2f\n", recno, item, total
}
END {
printf "--------------------------\n"
printf "\tTotal Rs. %6.2f\n" ,gtotal
printf "==========================\n"
}
```

Run it as

$ awk -f bill4 inven

Bill for the 4-March-2001.
By Vivek G Gite.
---
1 Pen Rs. 100.00
2 Pencil Rs. 20.00
3 Rubber Rs. 10.50
4 Cock Rs. 91.00

Total Rs. 221.50
==========

From the above output you can clearly see that printf can format the output. Let's try to understand formatting of printf statement. For e.g. %2d, number between % and d, tells the printf that assign 2 spaces for value. Same way if you write following awk program

```
$ cat > prf_demo
{
na = $1
printf "|%s|", na
printf "|%10s|", na
printf "|%-10s|", na
}
```

Run it as follows (and type the **God**)

$ awk -f prf_demo
God
|*God*|
| *God*|
|*God* |
(press CTRL + D to terminate)

if condition in awk

General syntax of if condition is as follows:

Syntax:

```
if ( condition )
{
```

```
                Statement 1
                Statement 2
                Statement N
                if condition is TRUE
        }
        else
        {
                Statement 1
                Statement 2
                Statement N
                if condition is FALSE
        }
```

Above if syntax is selfexplontary, now lets move to next awk program

```
$ awk > math2
BEGIN {
myprompt = "(To Stop press CTRL+D) > "
printf "Welcome to MyAddtion calculation awk program v0.1\n"
printf "%s" ,myprompt
}
{
no1 = $1
op = $2
no2 = $3
ans = 0
if ( op == "+" )
{
ans = $1 + $3
printf "%d %c %d = %d\n" ,no1,op,no2,ans
printf "%s" ,myprompt
}
else
{
printf "Opps!Error I only know how to add.\nSyntax: number1 +
number2\n"
printf "%s" ,myprompt
}
}
END {
printf "\nGoodbuy %s\n" , ENVIRON["USER"]
}
```

Run it as follows (Give input as **5 + 2** and **3 - 1** which is shown in bold words)
$awk -f math2

Welcome to MyAddtion calculation awk program v0.1
(To Stop press CTRL+D) > ***5 + 2***
5 + 2 = 7
(To Stop press CTRL+D) > ***3 - 1***

Opps!Error I only know how to add.

Syntax: number1 + number2

(To Stop press CTRL+D) >
Goodbuy vivek

Loops in awk

For loop and while loop are used for looping purpose in awk. Syntax of for loop

Syntax:

```
for (expr1; condition; expr2)
{
        Statement 1
        Statement 2
        Statement N
}
```

Statement(s) are executed repeatedly UNTIL the condition is true. BEFORE the first iteration, expr1 is evaluated. This is usually used to initialize variables for the loop. AFTER each iteration of the loop, expr2 is evaluated. This is usually used to increment a loop counter.

Example:

```
$ cat > while01.awk
BEGIN{
printf "Press ENTER to continue with for loop example from LSST
v1.05r3\n"
}
{
sum = 0
i = 1
for (i=1; i<=10; i++)
{
sum += i; # sum = sum + i
}
printf "Sum for 1 to 10 numbers = %d \nGoodbuy!\n\n", sum
exit 1
}
```

Run it as follows:

$ awk -f while01.awk

Press ENTER to continue with for loop example from LSST v1.05r3
Sum for 1 to 10 numbers = 55
Goodbuy

Above for loops prints the sum of all numbers between 1 to 10, it does use very simple for loop to achieve this. It take number from 1 to 10 using i variable and add it to sum variable as sum = previous sum + current number (i.e. i).

Consider one more example of for loop:

```
$ cat > for_loop
BEGIN {
printf "To test for loop\n"
printf "Press CTRL + C to stop\n"
}
{
```

```
for(i=0;i<NF;i++)
{
printf "Welcome %s, %d times.\n" ,ENVIRON["USER"], i
}
}
```

Run it as (and give input as **Welcome to Linux!**)
$ awk -f for_loop
To test for loop
Press CTRL + C to Stop
Welcome to Linux!
Welcome vivek, 0 times.
Welcome vivek, 1 times.
Welcome vivek, 2 times.
LSST v1.05 > Chapter 7 > Loops in awk

You can use while loop as follows:

Syntax:

```
while (condition)
{
        statement1
        statement2
        statementN
        Continue as long as given condition is TRUE
}
```

While loop will continue as long as given condition is TRUE. To understand the while loop lets write

```
$ cat > while_loop
{
no = $1
remn = 0
while ( no > 1 )
{
remn = no % 10
no /= 10
printf "%d" ,remn
}
printf "\nNext number please (CTRL+D to stop):";
}
```

Run it as
$awk -f while_loop
654
456
Next number please(CTRL+D to stop):587
785
Next number please(CTRL+D to stop):

sed - Quick Introduction

SED is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). SED works by making only one pass over the input(s), and is consequently more efficient. But it is SED's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

$ sed 's/Linux/UNIX(system v)/' demofile1
Hello World.
This is vivek from Poona.
I love linux.
.....
...
.....
linux is linux

General Syntax of sed

Syntax:

sed -option 'general expression' [data-file]
sed -option sed-script-file [data-file]

sed utility - Editing file without using editor

For this part of tutorial create data file as follows
teaormilk
India's milk is good.
tea Red-Lable is good.
tea is better than the coffee.

After creating file give command
$ sed '/tea/s//milk/g' teaormilk > /tmp/result.tmp.$$
$ cat /tmp/result.tmp.$$
India's milk is good.
milk Red-Lable is good.
milk is better than the coffee.

sed utility is used to find every occurrence of tea and replace it with word milk. sed - Steam line editor which uses 'ex' editors command for editing text files without starting ex. (Cool!, isn't it? no use of text editor to edit anything!!!)

Syntax:

sed {expression} {file}

Use of sed utility: sed is used to edit (text transformation) on given stream i.e a file or may be input from a pipeline.

Removing duplicate lines using uniq Utility

Create text file personame as follows:
personame
Hello I am vivek
12333
12333
welcome
to
sai computer academy, a'bad.
what still I remeber that name.
oaky! how are u luser?
what still I remeber that name.

After creating file, issue following command at shell prompt
$ uniq personame
Hello I am vivek
12333
welcome
to
sai computer academy, a'bad.
what still I remeber that name.
oaky! how are u luser?
what still I remeber that name.

Above command prints those lines which are unique. For e.g. our original file contains 12333 twice, so additional copies of 12333 are deleted. But if you examine output of uniq, you will notice that 12333 is gone (Duplicate), and "what still I remeber that name" remains as its. Because the uniq utility compare only adjacent lines, duplicate lines must be next to each other in the file. To solve this problem you can use command as follows

$ sort personame | uniq
General Syntax of uniq utility:
Syntax:
uniq {file-name}

Finding matching pattern using grep utility

Create text file as follows:
demo-file
hello world!
cartoons are good
especially toon like tom (cat)
what
the number one song
12221
they love us
I too

After saving file, issue following command,
$ grep "too" demofile

cartoons are good
especially toon like tom (cat)

I too grep will locate all lines for the "too" pattern and print all (matched) such line on-screen. grep prints too, as well as cartoons and toon; because grep treat "too" as expression. Expression by grep is read as the letter **t** followed by **o** and so on. So if this expression is found any where on line its printed. grep don't understand words.

Syntax:

grep "pattern" {file-name}

Basic concepts

A regular expression, often called a *pattern*, is an expression that describes a set of strings. They are usually used to give a concise description of a set, without having to list all elements.

the following operations to construct regular expressions.

alternation

A vertical bar separates alternatives. For example, "gray|grey" matches *gray* or *grey*.

grouping

Parentheses are used to define the scope and precedence of the operators. For example, "gray|grey" and "gr(a|e)y" are different patterns, but they both describe the set containing *gray* and *grey*.

quantification

A quantifier after a character or group specifies how often that preceding expression is allowed to occur. The most common quantifiers are ?, *, and +:

?

The question mark indicates there is **0 or 1** of the previous expression. For example, "colou?r" matches both *color* and *colour*.

*

The asterisk indicates there are **0, 1 or any number** of the previous expression. For example, "go*gle" matches *ggle*, *gogle*, *google*, etc.

+

The plus sign indicates that there is **at least 1** of the previous expression. For example, "go+gle" matches *gogle*, *google*, etc. (but not *ggle*).

Traditional Unix regular expressions

The "basic" Unix regular expression syntax is now defined as obsolete by POSIX, but is still widely used for the purposes of backwards compatibility. Most regular-expression Unix utilities, for example grep and sed, use it by default.

. Matches any single character

[ ] Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] matches any lowercase letter. These can be mixed: [abcq-z] matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [a-cq-z].

The '-' character should be literal only if it is the last or the first character within the brackets: [abc-] or [-abc]. To match an '[' or ']' character, the easiest way is to make sure the closing

bracket is first in the enclosing square brackets: []][ab] matches ']', '[', 'a' or 'b'.

[^] Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter. As above, these can be mixed.

^ Matches the start of the line (or any line, when applied in multiline mode)

$ Matches the end of the line (or any line, when applied in multiline mode)

\(\) Define a "marked subexpression". What the enclosed expression matched can be recalled later. See the next entry, \n. Note that a "marked subexpression" is also a "block"

\n Where n is a digit from 1 to 9; matches what the nth marked subexpression matched. This construct is theoretically irregular and has not been adopted in the extended regular expression syntax.

***** A single character expression followed by "*" matches zero or more copies of the expression. For example, "[xyz]*" matches "", "x", "y", "zx", "zyx", and so on.

\n*, where n is a digit from 1 to 9, matches zero or more iterations of what the nth marked subexpression matched. For example, "\(a.\)c\1*" matches "abcab" and "abcaba" but not "abcac".

An expression enclosed in "\(" and "\)" followed by "*" is deemed to be invalid. In some cases (e.g. /usr/bin/xpg4/grep of SunOS 5.8), it matches zero or more iterations of the string that the enclosed expression matches. In other cases (e.g. /usr/bin/grep of SunOS 5.8), it matches what the enclosed expression matches, followed by a literal "*".

\{x,y\} Match the last "block" at least x and not more than y times. For example, "a\{3,5\}" matches "aaa", "aaaa" or "aaaaa". Note that this is not found in some instances of regex.

Old versions of grep did not support the alternation operator "|".

Examples:

".at" matches any three-character string like hat, cat or bat
"[hc]at" matches hat and cat
"[^b]at" matches all the matched strings from the regex ".at" except bat
"^[hc]at" matches hat and cat but only at the beginning of a line
"[hc]at$" matches hat and cat but only at the end of a line

POSIX extended regular expressions are similar in syntax to the traditional Unix regular expressions, with some exceptions. The following metacharacters are added:

\+ Match the last "block" one or more times - "ba+" matches "ba", "baa", "baaa" and so on

? Match the last "block" zero or one times - "ba?" matches "b" or "ba"

| The choice (or set union) operator: match either the expression before or the expression after the operator - "abc|def" matches "abc" or "def".

Also, backslashes are removed: \{...\} becomes {...} and \(...\) becomes (...)

Examples:

"[hc]+at" matches with "hat", "cat", "hhat", "chat", "hcat", "ccchat" etc.

"[hc]?at" matches "hat", "cat" and "at"
"([cC]at)|([dD]og)" matches "cat", "Cat", "dog" and "Dog"

Since the characters '(', ')', '[', ']', '.', '*', '?', '+', '^' and '$' are used as special symbols they have to be escaped if they are meant literally. This is done by preceding them with '\' which therefore also has to be escaped this way if meant literally.
Examples:

"a\.(\(|\))" matches with the string "a.)" or "a.("

Examples: Wildcards

The File for These Examples	Wildcards #1	Wildcards #2	repetition
>cat file big bad bug bag bigger boogy	>grep "b.*g" file big bad bug bag bigger boogy	>grep "b.*g." file bigger boogy	>grep "ggg*" file bigger

Extended Regular Expression Syntax

We now discuss egrep syntax as opposed to grep syntax. Ironically, despite the origin of the name (extended), egrep actually has less functionality as it is designed for compatibility with the traditional egrep. A better way to do an extended "grep" is to use grep -E which uses extended regular expression syntax without loss of functionality.

grep	grep -E	Available for egrep?
a\+	a+	Yes
a\?	a?	Yes
expression1|expression2	expression1|expression2?	Yes
\(expression\)	(expression1)	Yes
\{m,n\}	{m,n}	No
\{,n\}	{,n}	No

\{m,}	{m,}	No
\{m}	{m}	No

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:].** For example, **[[:alnum:]]** means **[0-9A-Za-z]**, except the latter form depends upon the C locale and the ASCII character encoding, whereas the former is independent of locale and character set. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket list.) Most metacharacters lose their special meaning inside lists. To include a literal **]** place it first in the list. Similarly, to include a literal **^** place it anywhere but first. Finally, to include a literal **-** place it last.

regular expression may be followed by one of several repetition operators:

?	The preceding item is optional and matched at most once.
*	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.
{*n*}	**T**he preceding item is matched exactly *n* times.
{*n*,}	The preceding item is matched *n* or more times.
{*n*,*m*}	The preceding item is matched at least *n* times, but not more than *m* times.

nice

nice—Changes process priority
nice adds inc to the priority for the calling PID. Note that only the superuser may specify a negative increment, or priority increase. Note that internally, a higher number is a higher priority. Do not confuse this with the priority scheme as used by the nice interface.
To start such a process with a lower priority, just type nice before the name of the command. This will launch the command in a nice state:

nice dobigjob

With nice, it is possible to set the priority a process executes. Priorities range from 19 (the nicest program) to -20 (the highest priority). Only root can assign a high (negative) priority to a process.
Nice is set up to add 10 to the current priority of any process that by default is zero.
You can also specify a priority by providing the -n flag and a priority setting followed by the PID:

nice -n 12 dobigjob

Nice only works on jobs that have not started. To make an existing process nice, you use renice. If you want to give the Web server priority before all other processes

Scheduling Personal Reminders and Tasks with the 'at' Command

With the **at** command, you can execute commands **at** a specified time. Instead of placing a job immediately in the background, you can specify a time when you want it executed. You can then log out and the system will keep track of what command to execute and when to execute them.

The **at** command takes as its argument a time when you want commands executed. The time is a number specifying the hour followed by the keywords a.m. or p.m. you can also add a time. If no date is specified, today's date is assumed. The **at** command will then read in Linux Commands from the standard input. You can enter command **at** the keyboard, ending the standard input with a **CTRL + D**. You can also enter the commands into the file, which you can then redirect through the standard input to the **at** command.

Ex.

A command is execute at 4.00 am.

```
at  4am
ls –l
^D
```

The user places several commands in a file called latecmds and them redirects the contents of this file as input to an at command. The at command will execute the commands at 6.00 pm.

```
at 6pm < latecmds
```

If you need to keep track of important schedules, set reminders, or run programs unattended, you can use the at command. This command, found under the /usr/bin directory, schedules commands, or jobs, to be run at a time you specify.

For example, to send a pop-up reminder to your screen at the appropriate time, use the following:

```
$ at 16:15
at> ls –l
at> <EOT>
warning: commands will be executed using /bin/sh
job 12 at 1998-12-18 16:15
```

This tells the at command to run the long list of current directory. The endof-text (EOT) in the listing means you should press Ctrl+D to close the command to enter the job. If you make a mistake in the syntax of the command, you receive a mail message at the scheduled time.

You can also type these commands into a text file called carpool and run the commands by using the –f option, followed by the name of the file, with the following:

```
# at 16:15 -f carpool
# at –f carpool now + 5 hours

job 13 at 1998-12-18 10:19
```

The at command reads the commands from your text file and responds with a confirmation.

Using at.allow and at.deny

There are two access control files designed to limit which users can use the at facility. The file /etc/at.allow contains a list of users that are granted access, and the file /etc/at.deny contains a similar list of those who may not submit at jobs. If neither file exists, only the superuser is granted access to at. If a black /etc/at.deny file exists (as in the default configuration), all users are allowed to utilize the at facility.

Specifying times in an at job

at now	The job is run immediately.
at now +2 minutes	The job will start 2 minutes from the current time.
at now +1 hour	The job will start one hour from the current time.
at now +5 days	The job will start five days from the current time.
at now +4 weeks	The job will start four weeks from the current time.
at now next minute	The job will start in exactly 60 seconds.
at now next hour	The job will start in exactly 60 minutes.
at now next day	The job will start at the same time tomorrow,
at now next month	The job will start on the same day and at the same time next month.
at now next year	The job will start on the same date and at the same time next month.
at now next fir	The job will start at the same time next Friday.
at teatime	The job will run at 4 pm. They keywords **noon** and **midnight** can also be used.
at 16:00 today	same as previous
at 16:00 tomorrow	The job will run at 4 pm tomorrow.
at 2:45 p.m.	The job will run at 2:25 pm on the current day.
at 14:25	same as previous

at 5:00 sep 14 06	The job will begin at 5 am on September 14, 2006
at 5:00 9/14/06	same as previous
at 5:00 14.9.06	same as previous

Viewing scheduled jobs

Use the atq command to see a list of all your scheduled jobs, like this:

```
# atq
14 1998-12-18 12:00 a
15 1998-12-18 13:00 a
16 1998-12-18 14:00 a
17 1998-12-18 15:00 a
18 1998-12-25 16:15 a
```

This shows that four jobs are scheduled for December 18 with another scheduled for December 25. When you schedule jobs with the at command, a shell script containing each command is created in the /var/spool/atjobs directory. The atq command looks in this directory for your jobs and then prints them to your display.

The two most common queues names are "a" which represents the at queue and "b" represents the batch queue. All other letters (upper and lower case) can be used to specify queues with even lower priority levels. If the **atq** command lists the queue name as "=", this indicates that the job is currently running.

Deleting scheduled jobs

If you decide that you'd like to cancel a particular job, you can use the atrm command (equivalent to at –d) with the job number(or more than one) as reported by the atq command.

```
atrm 19 20
```

Using the batch command

If system resources are at a premium on your machine, or if the job you submit can run at a priority lower than normal, the batch command (equivalent to at –q b) may be useful. It is controlled by the same atd daemon and it allows job submissions in the same format as at submissions (although the time specification is optional).

Cron

Cron addresses the need to run commands periodically or routinely (at least, more often than you'd care to manually enter them) and allows considerable flexibility in automating the execution of the command. There are two access control files designed to limit which uses can use cron. The file /etc/cron.allow contains a list of users that are granted access, and the file /etc/cron.deny contains a similar list of those who may not submit cron jobs. If neither file exists, all users are granted access to cron.

There are four places where a job can be submitted for execution by the cron daemon crond:

1. The /var/spool/cron/crontabs/username file. This method, where each individual user controls his or her own separate file, is the method used on UNIX System V systems.
2. The /etc/crontab file. This is referred to as the system crontab file, and was the original crontab fiel from BSD UNIX and its derivatives. Only root has permission to modify this file.
3. The /etc/cron.d directory. Each file placed in this directory has the same format as the /etc/crontab file. Only root has permission to create or modity the files in this directory.

4. The /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, and /etc/cron.monthly directories. Each file in these directories is a shell script that runs at the times specified in the /etc/crontab file (by default at one minite after the hour, at 4:02 a.m. every day, Sunday at 4:22 am and 4:42 am on the first day of the month, respectively). Only root has permission to create or modity the file in these directories.

The standard format of an entry in the /var/spool/cron/username file consists of five fields specifying when the command should run: minute, hour, day, of the month, month, and day of the week. The sixth field is the actual command to be run.

The files in the /etc/cron.d directory and the /etc/crontab file use the same first five fields to determine when the command should run. The sixed field represents the name of the user submitting the job (because it cannot be inferred by the name of the file as in a /var/spool/cron/username directory), and the seventh field is the command to be run.

Valid /etc/crontab Field Values

Field Number	Field	Acceptable Values
1	Minute	Any integer between 0 to 59
2	Hour	Any integer between 0 to 23
3	Day of the month	Any integer between 0 to 31
4	Month	Ant integer between 0 to 12, or an abbreviation for the name of the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec)
5	Day of the week	Any integer between 0 to 7 (where both 0 and 7 can represent Sunday), or abbreviation for the day (Sun, Mon, Tue, Wed, Thu, Fri, Sat)
6	Command	Type the command to be execute

An asterisk (*) in any field indicates all possible values for that field. For example, an asterisk in the second column in equivalent to 0,1,2,……,22,23, and an asterisk in the fourth column means Jan, Feb, Mar,……, Nov, Dec. In addition, lists of values, ranges of values, and increments can be used.

Example , can use values in first column

45
3,6,76
10-30

Modifying Schedule Tasks with crontab

The files in /var/spool/cron should not be edited directly. They should only be accessed via the crontab command. To list the current contents of your own personal crontab file, type the following commands.

crontab -l

All crontab entire can be removed with the following command:

crontab -r

Ever if your personal crontab file does's exist, you can use the following command to being editing it:

crontab -e

The file will automatically open in text editor that is defined in your EDITOR or VISUAL environment variable. With vi as the default. When you're done, simply exit the editor , and (provided there were no syntax erros) your crontab file will be installed.

The root user can access any user's individual crontab file by using the -u username option to the crontab command.

Sort

Write sorted concatenation of all FILE(s) to standard output.

Sort [option] <file>

-f fold lower case to upper case characters
-n compare according to string numerical value
-r reverse the result of comparisons
-t “ch” use SEP instead of non- to whitespace transi-tion

+<n> define colume number start by 0

ex.

```
sort student1
sort -t “|” student1
sort -t “|” +2 student1
sort -t “|” +2 -n student1
```

PS

ps - report process status

ps [option]

ps gives a snapshot of the current processes.

-A select all processes
-a select all with a tty except session(bash) leaders
-e select all processes

sleep

sleep - delay for a specified amount of time

```
sleep number[suffix]
sleep [option]
```

Pause for NUMBER seconds. SUFFIX may be ‘s’ for seconds (the default), ‘m’ for minutes, ‘h’ for hours or ‘d’ for days. Unlike most implementations that require NUMBER be an integer, here NUMBER may be an arbitrary floating point number.

Touch

touch - change file timestamps

Update the access and modification times of each FILE to the current time.

-a change only the access time
-d parse STRING and use it instead of current time
-m change only the modification time
-t STAMP use [[YY]MMDDhhmm[.ss] instead of current time

ex.

touch -t "0506011106" student
touch -d "05-06-04" student

wc

wc - print the number of bytes, words, and lines in files

wc [option] <file>

Print byte, word, and newline counts for each FILE, and a total line if more than one FILE is specified. With no FILE, or when FILE is -, read standard input.

-c print the byte counts
-m print the character counts
-l print the newline counts
-L print the length of the longest line
-w print the word counts

file

file - determine file type

file tests each argument in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests, and language tests. The first test that succeeds causes the file type to be printed.

The type printed will usually contain one of the words text (the file contains only printing characters and a few common control characters and is probably safe to read on an ASCII terminal), executable (the file contains the result of compiling a program in a form understandable to some UNIX kernel or another), or ddaattaa meaning anything else (data is usually 'binary' or non-printable). Exceptions are well-known file formats (core files, tar archives) that are known to contain binary data. When modifying the file usr share magic or the program itself, preserve these keywords .. People depend on knowing that all the readable files in a directory have the word ''text'' printed. Don't do as Berkeley did and change ''shell commands text'' to 'shell script''. Note that the file usr share magic is built mechanically from a large number of small files in the subdirectory Magdir in the source distribution of this program.

Factor

Factor numbers

Factor [number]

Print the prime factors of each NUMBER

Units

units - unit conversion program

The 'units' program converts quantities expressed in various scales to their equivalents in other scales. The 'units' program can handle multiplicative scale changes as well as nonlinear conversions such as Fahrenheit to Celsius.

The units are defined in an external data file. You can use the extensive data file that comes with this program, or you an provide your own data file to suit your needs.
At the 'You have:' prompt, type the quantity and units that you are converting from. For example, if you want to convert ten meters to feet, type '10 meters'. Next, 'units' will print 'You want:'. You should type the type of units you want to convert to. To convert to feet, you would type 'feet'.

If you try to convert grains to pounds, you will see the following:

```
You have: meters
You want: foot
 * 3.2808399
 / 0.3048

You have: kg
You want: g
 * 1000              -- in 1 kg is 1000 g
 / 0.001             -- in 1 g is 0.001 kg
```

bc

bc - An arbitrary precision calculator language

bc is a language that supports arbitrary precision numbers with interactive execution of statements. There are some similarities in the syntax to the C programming language. A standard math library is available by command line option. If requested, the math library is defined before processing any files. bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read. (If a file contains a command to halt the processor, bc will never read from the standard input.)

The numbers are manipulated by expressions and statements. Since the language was designed to be interactive, statements and expressions are executed as soon as possible. There is no "main" program. Instead,code is executed as it is encountered.

- expr

The result is the negation of the expression.

++ var

The variable is incremented by one and the new value is the result of the expression.

-- var

The variable is decremented by one and the new value is the result of the expression.

var ++

The result of the expression is the value of the variable and then the variable is incremented by one.

var –

The result of the expression is the value of the variable and then the variable is decremented by one.

expr + expr

The result of the expression is the sum of the two expressions.

expr – expr

The result of the expression is the difference of the two expressions.

expr * expr

The result of the expression is the product of the two expressions.

expr / expr

The result of the expression is the quotient of the two expressions. The scale of the result is the value of the variable Scale.

expr % expr

The result of the expression is the "remainder" and it is computed in the following way. To compute a%b, first a/b is computed to Scale digits. That result is used to compute a-(a/b)*bto the scale of the maximum of scale+scale(b) and scale(a). If Scale is set to zero and both expressions are integers this expression is the integer remainder function.

expr ^ expr

The result of the expression is the value of the first raised tothe second. The second expression must be an integer. (If the second expression is not an integer, a warning is generated and the expression is truncated to get an integer value.) The scaleof the result is scale if the exponent is negative. If the exponent is positive the scale of the result is the minimum of the scale of the first expression times the value of the exponent and the maximum of scale and the scale of the first expression. (e.g. scale(a^b) = min(scale(a)*b, max(scale, scale(a))).) It should be noted that expr^0 will always return the value 1

(expr)

This alters the standard precedence to force the evaluation of the expression.

var = expr

The variable is assigned the value of the expression.

var <op>= expr

This is equivalent to "var = var <op> expr" with the exception that the "var" part is evaluated only once. This can make a difference if "var" is an array.

Relational expressions are a special kind of expression that always evaluate to 0 or 1, 0 if the relation is false and 1 if the relation is true. These may appear in any legal expression. (POSIX bc requires that relational expressions are used only in if, while, and for statements and that only one relational test may be done in them.) The relational operators are

expr1 < expr2
The result is 1 if expr1 is strictly less than expr2.

expr1 <= expr2
The result is 1 if expr1 is less than or equal to expr2.

expr1 > expr2
The result is 1 if expr1 is strictly greater than expr2.

expr1 >= expr2
The result is 1 if expr1 is greater than or equal to expr2.

expr1 == expr2
The result is 1 if expr1 is equal to expr2.

expr1 != expr2
The result is 1 if expr1 is not equal to expr2.

Boolean operations are also legal. (POSIX bc does NOT have Boolean operations). The result of all boolean operations are 0 and 1 (for false and true) as in relational expressions. The Boolean operators are:

!expr
The result is 1 if expr is 0.

expr && expr
The result is 1 if both expressions are non-zero.

expr || expr
The result is 1 if either expression is non-zero.

The expression precedence is as follows: (lowest to highest)
|| operator, left associative
&& operator, left associative
! operator, nonassociative

Relational operators, left associative Assignment operator, right associative
+ and - operators, left associative
*, / and % operators, left associative
^ operator, right associative
unary - operator, nonassociative
++ and -- operators, nonassociative

This precedence was chosen so that POSIX compliant b□bc□c programs will run correctly. This will cause the use of the relational and logical operators to have some unusual behavior when used with assignment expressions. Consider the expression:

a = 3 < 5

Most C programmers would assume this would assign the result of "3 < 5" (the value 1) to the variable "a". What this does in bc is assign the value 3 to the variable "a" and then compare 3 to 5. It is best to use parenthesis when using relational and logical operators with the assignment operators.

Banner

The banner command creates posters by blowing up its argument on the screen. On each line it can display a minimums of ten characters. There are no options, and it is simple to use.

```
banner Aryan
```

When the arguments are used within quotes, the display can be formatted suitably: when you use the pair of single quotes, along with two spaces in the first argument, the message is printed in two “lines” with a suitably offset. You can also use double quotes if you like, but make sure that you don't use the characters $ and ` (back quote) in the argument.

```
Banner '    Aryan'  'College'
```

lpr

with the lpr command, you can print document to both local and remote printers. Document files can be either added to the end of the lpr command line or directed to the lpr command-using pipe (|).

```
lpr doc1.txt
```

when you just specify a document file with lpr, output is directed to the directed printer. As an individual user, you can change your default printer by setting the value of the PRINTER variable. Typically, you would add the PRINTER variable to one of your startup file, such as ~/.bashrc.

```
export PRINTER=lp3
```

To override the default printer, specify a particular printer on the lpr command line. The following example uses the –P option to select a different printer.

```
lpr –P Epson doc1.txt
```

The lpr command has a variety of options that allow lpr to interpret and format several different types of documents. The following options that you can use one the command line and the types of text those options can process.

-m Sends mail after the job has been printer.
-r Removes the file after it has been printer.
-s Uses a symbolic link to obtain the file to print rather than copying the file to a spool directory. This prevents very large print jobs from taking up space in the print queue. If you do this, however, you must be sure not to delete the original file until after it has been printer.
-#n Replace the n with a number representing the number of copies you want to print.

Removing print jobs with lprm

Users can remove their own print jobs from the queue with the lprm command. Used alone on the command line, lprm will remove all the user's print jobs from the default printer. To remove jobs from a specific printer, use the –P option as follows:

```
lrm –P lp0
```

To remove all print jobs for the current user, type the following

```
lprm –
```

The root user can remove all the print jobs for a specific user by indicating that user on the lprm command line.

```
lprm student
```

To remove an individual print job from the queue, indicate the job number of that print job on the lprm command line. To first find out what the job number is, type the lpq command with the –l option.

```
lpq –l
```

To remove that job, type the following

```
lprm 008anov1kj
```

Displaying queue information with lpq

You can list the jobs waiting to be printed with the lpq command.

```
lpq

lp0 is ready and printing
Rank   Owner Job     Files                  Total Size
active  root            10      /tmp/out.ps     13887 bytes
1st             student 11      house.txt               546 bytes
2nd             alok            12      memo.ps                 384489 bytes
```

To see details about your print jobs, use the –l option with lpq as

```
lpq –l

lp0 is read and printing
root : active                   [job 010AudWpyb]
        /tmp/out.ps             13887 bytes
```

Controlling printers with lpc

The lpc command lets you check the status of printers, enable and disable printers, and change the arrangement of jobs in the queue. Any action you do with the lpc command can be done either to an individual printer (by indicating a printer name) or to all printers (by using the word all). To disable a specific printer, use the disable option

```
lpc disable Epson
```

Epson :
Queuing disable

To enable a disabled printer, use the enable option and type the following:

lpc enable Epson

Epson:
Queuing enabled

When a printer is disabled, it can still continue to receive print jobs and hold them in the queue until the printer is enabled again. If you want to prevent jobs from being spooled at all, use stop option as follows:

lpc stop Epson

Epson:
Printing disabled

To start the printing daemon again and resume printing, use the start option as follows:

lpc start Epson

Epson:
Printing enabled
Daemon started

To disable a printer and kill the lpd-printing daemon (so that no further jobs can be printed), use the abort option as follows:

lpc abort Epson

Epson:
Printing disabled
Daemon (pid 1989) killed

To restart the lpd daemon after it has been stopped, use the restart option as follows:

lpc restart Epson

lp0:
no daemon to about
lp0:
daemon started

Understanding System Administrations

Even if you are the only person using a Red Hat Linux system, system administration is still set up to be separate from other computer use. To do most tasks, you need to be logged in as the root user (also referred to as the super user). Other users cannot change, or in some cases, even see some of the configuration information for a Red Hat Linux system. In particular, security features such as passwords are protected from general view.

Administrative Configuration files

Configuration files are another mainstay of Red Hat Linux administration. They contain the information that defines who can use your computer, what programs are run automatically, and which computers can connect to your system. Almost everything you set up for your particular computer- user accounts, network addresses, or GUI preferences – are stored in plain-text files, which has some advantages and some disadvantages.

This advantage of plain-text files is that it is easy to read and change them. Any text editor will do. On the downside, however, is that as you edit configuration files, no error checking occurs. You have to run the program that reads these files (such as a network daemon or the X GUI) to determine if you set up the files correctly. A comma or a quote in the wrong place can sometimes cause a whole interface to fail.

In terms of a general perspective of configuration files, however, there are several locations in the Red Hat Linux file system that configuration files are stored. Here are some of the major locations.

- $HOME – All users store information in their home directories that directs how their login accounts behave. Most configuration files begin with a dot (.), so they don't appear in a user's directory when your use a standard ls command (you need to type ls –a to see them). There are dot files that define how each user's shell behaves, the look and feel of the desktop, and what options are used with your text editor. There are even files (such as .rhosts) that configure network permissions for each user.
- /etc – This contains many of the most basic Red Hat Linux system configuration files. Files include:
    - aliases – Can contain distribution lists used by the Red Hat Linux mail service.
    - bashrc – Sets systemwide defaults for bash shell users. (By default, it only sets the shell prompt to include the current user name, host name, and current directory.)
    - crontab – Sets cron environment and times for running automated tasks.
    - csh.cshrc (or cshrc) – Sets systemwide defaults for csh (C shell) uses.
    - exports – Contains a list of local directories that are available to be shared by remote computers using the Network File System (NFS).
    - fdprm – Sets parameters for common floppy disk formats.
    - fstab – Identifies the devices for common storage media (hard disk, floppy, cdrom, etc.) and locations where they are mounted in the linux system. This is used by the mount command to choose which file systems to mount.
    - gettydefs – Contains line definitions used by terminals devices (modem, dumb terminals, and remote login over terminal devices).
    - group - Identifies group names of group ids (GIDs) that are defined on the systems. Group permissions in Linux are defined by the second of three sets of rwx (Read, Write, Execute) bits associated with each file and directory.
    - host.conf – Sets the locations in which domain names (for example, redhat.com) are searched for on TCP/IP networks (such as the internet). By default, the local hosts file is searched, and then any nameserver entries in resolv.conf.
    - hosts – Contains IP addresses and hostname that you can reach from your computer. (Usually this file is just used to store names of computers on your LAN or larger private network.)
    - hosts.allow – Lists host computers that are allowed to use TCP/IP services from the local computer.
    - hosts.deny – Lists host computer that are not allowed to use TCP/IP services from the local computer.
    - inetd.conf – Contains a list of services and related information for TCP/IP services used by inetd. (When the inetd daemon process receives a request for a service, it uses the entries here to determine which daemon processes to start to handle the request.)
    - info-dir – Contains the top heading for information that is available from the info command.

- inittab – Contains information that defines what programs start and stop when Linux boots, shuts down, or goes into different states in-between.
- issue – Contains login lines that are displayed when a terminal is ready to let you login to Linux from a local terminal or the console in text mode.
- issue-net – Contains login lines that are displayed to users that try to login to the Linux system from a computer on the network using the telnet service.
- lilo.conf – Sets various parameters used by the Linux boot loader (lilo) to boot your Linux system.
- mail.rc – Sets systemwide parameters associated with using mail.
- man.config – Used by the man command to determine the default path to the location of man pages.
- mtab – Contains a list of file systems that are currently mounted.
- passwd – Stores account information for all valid users for the system. Also includes other information, such as the home directory and default shell.
- profile – Sets systemwide environment and startup programs for all users. This file is read when the user logs in.
- protocols – Sets protocol numbers and names for a variety of internet services.
- redhat-release – Contains a string identifying the current Red Hat release number.
- resolv.conf – Identifies the location of DNS name-server computers that are used by TCP/IP to translate Internet host.domain names into IP addresses.
- rpc – Defines remote procedure call names and number.
- services – Defines TCP/IP services and their port assignments.
- shadow – Contains encrypted passwords for users that are defined in the passwd file. (This is viewed as a more secure way to store passwords then the original encrypted passwod in the passwd file. The passwd file needs to be publicly readable, while the shadow file can be unreadable by all but the root user.)
- shells – Lists the shell command-line interpreters (bash, sh, csh, etc.) that are available on the system, as well as their locations.
- syslogs.conf – Defines what logging messages are gathered by the syslogd daemon and what files they are stored in. (Typically, log messages are stored in files contained in the /var/log directory.)
- termcap – Lists definitions for character terminals, so that character based applications know which features are supported by a given terminal. Graphical terminals and application have made this file obsolete to most people. (Termcap was the BSD UNIX way of storing terminal information; UNIX System V used definitions in /usr/share/terminfo files.)

- /etc/X11 – Contains subdirectories that each contain systemwide configuration files used by X and different X Window managers that are available with Red Hat Linux. The XF86Config file (which makes your computer and monitor usable with X) and the configuration directories that contain the files that are used with xdm and xinit to start X are contained here.
  Directories relating to window managers contain files that include the default values that a user will get if the user start one of these window managers on your system. Window managers that may have systemwide configuration files in these directories include AnotherLevel, Fvwm, Fvm2, Gnome, and Twm.
  Some of the files and directories in /etc/X11 are linked to locations in the /usr/X11R6 directory structure.
- /etc/cron* - This set of directories contains files that define how the crond utility runs applications on a daily, hourly, monthly, and weekly schedule.

- /etc/default – Contains files that set defaults values for various utilities. For example, the files for the useadd command defines the default group number. Home directory, password expiration date, shell, and skeleton directory (/etc/skel) that are used when creating a new user account.
- /etc/httped – Contains files that are used to configure the behavior of your Apache server (specifically, the httped daemon process).
- /etc/pcmcia – Contains configuration files that enable you to configure a variety of PCMCIA cards for your computer. (PCMCIA slots are those opening on your laptop for attaching credit-card sized cards to your computer, such as modems and external CD-ROMS.)
- /etc/ppp – Contains configuration files that are used to set up Point-to-Point protocol (so that you can have computer dial out to the internet.)
- /etc/rc.d – Contains several subdirectories, each of which contains configuration files that direct the programs that are started and stopped during system startup, shutdown, and changes of system states. There is a separate subdirectory for each valid system state.
- /etc/security – Contains files that set a variety of default security conditions for your computer.
- /etc/skel – Any files contained in this directory are automatically copied to a user's home directory when that user is added to the system. By default, most of these files are dot (.) files, such as .Xdefaults (for setting X Window System defaults) and .bashrc (for setting default values used with the bash shell).
- /etc/sysconfig – Contains important system configurations files that are created and maintained by several Red Hat Linux applications (including netcfg and linuxconf).
- /etc/uucp - Contains configuration files used with Taylor UUCP (a nonstandard version of the uucp facility that is used to create modem, direct line, and other serial connections with other computers).

Administrative log files

One thing that Red Hat Linux does well is keep track of itself. You want to monitor your system to determine if people are trying to access your computer illegally. In any of these cases, you can use log files to help track down the problem.

The main utilities for logging error and debugging messages for Linux are the syslogd and the klogd daemons. General system logging is done by syslogd. Logging that is specific to kernel activity is done by klogd. Logging is done based on information in the /etc/syslog.conf file. Messages are typically directed to log files that are usually in the /var/log directory. Here are some common log files from that directory and the messages they contain:

- cron – Contains messages that are output by the cron command (which is used to run tasks at set times). Here you can see when tasks are started and any error conditions that may have occurred.
- secure – Contains messages that may indicate security breeches. Connections from remote hosts are logged, as are attempts to log in to your system.
- maillog – Activities of the sendmail daemon (which forwards e-mail to other computers) are logged in this file.
- messages – Messages associated with many daemon processes are directed to the messages file.

Besides these files, messages are also directed to several directories located in the /var/log directory. These directories include:

- httpd – Messages from the Apache Web server are logged in files in this directory.
- uucp – Messages from the Unix-to-Unix copy (uucp) facility, are stored here.

Other administrative logins

The administrative logins that are configured automatically for Linux systems are, by tradition, assigned UID numbers under 100. These logins have no passwords by default, so you can't use ant administrative login separately until you assign it a permissions of some executables, spool files, and log files to make them owned by the administrative login. The following is a list of the administrative logins that are configured automatically for Red Hat Linux:

- lp – This user can control some printing features. Having a separate lp administrator, enables someone other than the superuser to do such things as move or remove lp logs and print spool files. The home directory for lp a /var/spool/lpd.
- mail – This user can work with administrative e-mail features. The mail group has group permissions to use mail files in /var/spool/mail (which is also the mail user's home directory).
- uucp – This user owns various uucp commands (used for dial-up serial communications). It is the owner of log files in /var/log/uucp, spool files in /var/spool, administrative commands (such as uuchk, uucico, uuconv and uuxqt) in /usr/sbin, and user commands (uucp, cu, uuname, uustat, and uux) in /usr/bin. The home directory for uucp is /var/spool/uucp.
- bin – This usr owns many command in /bin in traditional UNIX systems. This is not the case in Red Hat Linux because root tends to own most executable files. The home directory of bin is /bin.
- news – This user could be used to do administration of Internet of news services, depending on how you set permission for /var/spool/news and other news-related resources. The home directory for news is /var/spool/news.

Getting to know Your Red Hat Linux System

Hostname and Linux Version

The uname command can print some basic information about your Red Hat Linux system, including:

- Operating system name (i.e. Linux)
- System's host name.
- Linux kernel release number
- Current date and time
- Processor type

The following is an example of the uname –a command (the –a prints all the information at once):

uname –a

Linux localhost 2.4.21-4.EL #1 Fri Oct 3 18:13:58 EDT 2003 i686 i686 i386 GNU/Linux

The system's host name is used to identify your computer (in particular, it is used by remote systems to contact you over a network). The Linux release number (2.4.21-4.EL) identifies release of the Linux kernel. Check this number to make sure you have the latest kernel. The other information includes the current date, time, and time zone, as well as the processor of your computer (i686).

You can permanently change your computer name using the hostname command as follows:

hostname <name>

The hostname is read automatically by different programs that need to indicate which computer is being used. dnsdomainname lists your domain name. If your computer uses NIS services, you can print your NIS domain name with the domainname command.

User Accounts

User login accounts are listed in the /etc/passwd file. You can list the contents of this file to see what users have accounts on your Linux System as follows:

more /etc/passwd

Administrative logins (up to UID 100 in the third colon-separated field) should make up most of the first few entries in this file. Regular user accounts (with UIDs above 100) are usually added in after the administrative accounts.

The Kernel

The heart of the Red Hat Linux system (actually, of any UNIX system) is the kernel. The kernel provides the interface between you (and the programs that you run) and the hardware (hard disks, RAM, network cards. Etc.). By using the /proc file system, you can discover a lot of information about your kernel by displaying the contents of /proc files.

For each process currently running in Red Hat Linux, there is a directory in /proc consisting of the process number for the running process. (Type ps aux | more to see the running processes and their associated PID numbers.) In the /proc directory, there are also other files that are connected to certain features (such as networking, SCSI devices, and other components).

cat version

The output of this command contains the Linux version number and other information (such as the compiler version and the system install date). There are other files under the /proc directory structure that you can also list to learn interesting information about your running Red Hat Linux system. Here are a few files that you can "cat" to get information:

- cpuinfo – Tells you the type of CPU in your computer, the speed (CPU MHz), the CPU family, and other information related to your computer's processor.
- devices – Displays the character and block devices that are currently being used on your computer, along with their major device numbers.
- ioports – Shows the I/O port addresses for the devices on your computer.
- meminfo – Contains information about memory usage and swap space usage. You can see the total amounts of memory and the how much is currently.
- modules – Shows a list of modules that are currently installed in the system.
- mounts – Displays the file systems that are currently mounted in the system.
- partitions – Contains the name of your hard disk partitions, the number of blocks in each partition, and each partition's major and minor device number.
- pci – lists the PCI devices that are installed in your computer. You can see the bus device numbers, names and other information. For cards that are installed (such Ethernet or modem cards), you can see their IRQs, addresses, and other information.
- swaps – Shows the swap partitions that are currently mounted on your system, along with its size and the amount of space being used.
- net/dev – Display the contents of the net/dev file to see your active network interfaces.
- sys/* - Look at the contents of these directories for information related to debugging (debug), devices (dev), file systems (fs), the kernel (kernel), networks (net), and processes (proc). The net directory contains some of the most useful information. Go to net/ipv4 and list the contents of these files to see when certain features are turned on or off.

Graphical Administrative Interfaces

Using linuxconf

The most complete graphical utility for working with Red Hat Linux is linuxconf. As a GUI interface, linuxconf isn't fancy. Don't expect to see lots of icons or to be able to drag-to-drop items on the display.

- Configuration and Control tools for many different features are all contained in one place. Just click on an activity in the left column and a from for configuring the item appears in the right column. You don't have to search blindly in /etc for the right files to edit.
- Some error checking is done. In many cases, linuxconf will prevent you from entering invalid values in the fields.
- Options are offered. If, for example, you are adding a network interface, you can click on a pull-down menu to select from the interfaces that Linux knows about (such as PPP or Ethernet).

Administrative activities are divided into two major categories in linuxconf: Config and Control. Config activities enable you to set up your network interfaces (for both client and server features), work with user accounts, configure file systems, and manage how Linux boots. Control activities enable you to work with feature that have already been configured, including starting and stopping services, mounting/unmounting file system, controlling the files and systems used by linuxconf, and working with system logs.

Linuxconf Configuration tasks

Under the configuration section in linuxconf are tasks for setting up your network, creating user accounts, working with file systems, initializing system services, and choosing boot modes, Networking tasks are divided into those that apply to your computer as a client and those that apply to your computer as a server.

linuxconf Networking Tasks

Client networking tasks let you view and configure information associated with your computer's hostname, the networks interfaces that are attached to your computer, and the routes that you can use to get to other hosts and network.

- Hostname - Your hostname is how other computers on the network identify your. It can contain the full hostname.domainname form.
- Adaptor – The network interfaces (i.e. Ethernet cards, PPP dial-up connections, etc) that give you access to the network can be viewed by clicking the Adaptor tabs on the Basic host information from.

The way that your system resolves Internet hostnames into IP addresses is by identifying DNS servers that can do name-to-address resolution. Click the Domain Name Server Specification (DNS) task to add your default domain and one or more name servers. You can also indicate which domains to search for addresses.

Under Routing and Gateways (in the configuration section) you can define how your networking requests are routed across gateway machines (those that are connected to your sub networks) to reach beyond your local network. You can also specify routes to other local area networks.

Other network services that you can configure include the Network Information Service (NIC), IPX interface, and serial IP connections (PPP, SLIP, and PLIP). NIS is a way of having a central server store the information that each client computer needs to start up. IPX is a networking interface protocol that is popular with Netware servers and PPP, SLIP and PLIP are ways to connect using Internet protocols across modems, direct connections, and other serial lines.

linuxconf control tasks

The control section of linuxconf lets you work with Red Hat Linux features that change the ongoing operation of your Red Hat Linux system. Here are some of the tasks that you can do from his section:

- Activate Configuration – For changes that you make to take effect, some services have to be stopped and restarted. This task checks what needs to be restart, based on the changes that you made, then restarts those services when you are ready.
- Shutdown/reboot – Use this task to either shutdown and halt your computer or to reboot it.
- Control service activity – Use this task to enable or disable a variety of network services.
- Mount/Unmount file systems – Any local or NFS file systems that you configured to be mountable (in the Configuration section) can be mounted or unmounted using these tasks.
- Configure superuser schedule – You can add commands that are run at a set schedule (using the cron facility) as the root user by adding entries under this task.
- Archive Configurations – With this task you can archive the configuration files that you set up so that you can recall these saved configuration files later. This task can be used to get you back to get you back to a state if your configuration files get wrecked.
- Switch system profile – You can recall a past archive of configuration files (and save the current configuration files) using this task.
- Control PPP/SLIP/PLIP – Your can activate any currently inactive PPP, SLIP or PLIP interface, or deactivate any active ones using this task.
- Control files and systems – Select tasks from this section to change the way configuration files, commands, file permissions, modules, system profiles, and linuconf add-ons are configured and used in Linux.
- Logs – Read logs of Linuxconf error messages or boot messages from this task.
- Date & time – change the date, time and time zone from this task.
- Features – Modify the keyboard mapping used for your Linux computer, the language, or several features associated with how HTML is used on your computer.

In current version of Linux (Redhat Enterprice Linux 3.0) is manipulate administrative tasks by using some utilities. These utilities command is locate in /usr/bin directories, the naming convension is **redhat-**
Example :

Configure your Apache Server

redhat-config-httpd

All these utilities is use in GUI environment, but some utilities is use in CUI environment.

Creating a file system on a disk or partition

It is possible to create a file system for any supported file system type on a disk or partition that you choose with the mkfs command.

```
mkfs –t ext3 /dev/fd0
```

The statistics that are output with the formatting done by the mkfs command. The number of inodes and block created are output. The number of block per group and fragments per group are also output.

Syntax

```
mkfs [option] <filesys>
```

Option

-V Produce verbose output, including all file system-specific commands that are executed. Specifying this option more than once inhibits execution of any file system-specific commands. This is really only useful for testing.
-f <fstype> Specifies the type of file system to be built. If not specified, the default file system type (currently ext3) is used.
-c Check the device for bad blocks before building the file system.
-l <filename> Read the bad blocks list from filename.

Configuring Modules

Red Hat Linux comes with tools for configuring the drivers that stand between the programs you run (such as CD players and Web browsers) and the hardware they use (CD−ROM drives and network cards). The intention is to have the drivers your system needs most often built into the kernel; these are called resident drivers. Other drivers that are added dynamically as needed are referred to as loadable modules.

Finding available modules

If you have installed the Linux kernel source code (kernel−source package), source code files for available drivers are stored in subdirectories of the /usr/src/linux−2.4/drivers directory. There are several ways of finding information about these drivers:

- **make xconfig —** With /usr/src/linux−2.4 as your current directory, type make xconfig from a Terminal window on the desktop. Select the category of module you are interested in and click Help next to the driver that interests you. The help information that appears tells you the module name and a description of the driver.
- **Documentation** — The /usr/src/linux−2.4/Documentation directory contains lots of plain−text files describing different aspects of the kernel and related drivers. Of particular interest is the modules.txt file (which describes how to work with modules) and the Configure.help file (which contains all the help files hardware drivers).
- **kernel−doc —** The kernel−doc software package (available on CD−2 of the Red Hat Linux distribution) contains a large set of documents describing the kernel and drivers. These documents are stored in the /usr/share/doc/kernel−doc* directory.

After modules have been built, they are installed in the /lib/modules/2.4* directory. The name of the directory is based on the current release number of the kernel. Modules that are in that directory can then be loaded and unloaded as they are needed.

Listing loaded modules

To see which modules are currently loaded into the running kernel on your computer, you can use the lsmod command. Here is an example:

```
# lsmod
```

Module	Size	Used	by
sr_mod	15120	0	(autoclean)
es1371	26784	0	(autoclean)
ac97_codec	8704	0	(autoclean) [es1371]
gameport	1920	0	(autoclean) [es1371]
soundcore	4112	4	(autoclean) [es1371]
binfmt_misc	6272	1	

nuscsitcp	17200	0	(unused)
autofs	10816	1	(autoclean)
tulip	46400	1	
ipchains	36960	0	(unused)
ide−scsi	8192	0	
scsi_mod	93568	3	[sr_mod nuscsitcp ide−scsi]
hid	18160	0	(unused)
input	3456	0	[hid]
usb−uhci	21440	0	(unused)
usbcore	50432	1	[hid usb−uhci]
ext3	50656	2	
jbd	39376	2	[ext3]

This output shows a variety of modules that have been loaded on a Linux system. The modules loaded on this system include several to support the Ensoniq 1371 sound card that is installed (es1371, ac97_codec, gameport, and soundcore). There are also modules to support the IDE CD−ROM drive that runs in SCSI emulation on this system (scsi_mod, sr_mod, nuscsitcp, and ide−scsi).

To find information about any of the loaded modules, you can use the modinfo command. For example, you could type the following:

```
# modinfo −d es1371
"ES1371 AudioPCI97 Driver"
```

Loading modules

You can load any module that has been compiled and installed (to the /lib/modules directory) into your running kernel using the insmod command. The most common reasons for loading a module are that you want to use a feature temporarily (such as loading a module to support a special file system on a floppy you want to access) or to identify module that will be used by a particular piece of hardware that could not be autodetected.

Here is an example of the insmod command being used to load the parport module. The parport module is used to provide the core functions for sharing a parallel port with multiple devices.

```
# insmod parport
Using /lib/modules/2.4.6−3.1/kernel/drivers/parport/parport.o
```

After parport is loaded you could load the parport_pc module to define the PC−style ports that are available through the interface. The parport_pc module lets you optionally define the addresses and IRQ numbers associated with each device sharing the parallel port. Here is an example:

```
# insmod parport_pc io=0x3bc irq=auto
```

A device is identified as having an address of 0x3bc. The IRQ for the device is auto−detected. The insmod command loads modules temporarily. At the next system reboot, the modules you enter disappear. To permanently add the module to your system, add the insmod command line to one of the start−up scripts that are run a boot time.

Removing modules

You can remove a module from a running kernel using the rmmod command. For example, to remove the module parport_pc from the current kernel, type the following:

```
# rmmod parport_pc
```

If the module is not currently busy, the parport_pc module is removed from the running kernel.

Reconfiguring Hardware with kudzu

When you add or remove hardware from your computer and reboot Red Hat Linux, a window appears during the reboot process advising that hardware has either been added or removed and asking if you want to reconfigure it. The program that detects and reconfigures your hardware is called kudzu.

The kudzu program is a hardware autodetection and configuration tool that runs automatically at boot time. If you like, you can also start kudzu while Red Hat Linux is running. In either case, here is what kudzu does:

1. It checks the hardware connected to your computer.
2. It compares the hardware it finds to the database of hardware information stored in the /etc/sysconfig/hwconf file.
3. It prompts you to change your system configuration, based on new or removed hardware that was detected.

The following is a list of hardware that kudzu can detect (according to the kudzu README file), followed by a description of what kudzu does to configure the device. Other devices may be detected as well (such as USB devices).

- Network devices — Adds an Ethernet interface alias (eth0, eth1, etc.) if necessary and either migrates the old device configuration or creates a new one.
- SCSI — Adds an alias for scsi_hostadapter.
- Video card — Runs the Xconfigurator command to configure the video card.
- Sound card — Runs the sndconfig command to configure and test the sound card.
- Mouse — Links the new mouse device to /dev/mouse and runs the mouseconfig command to configure and test the mouse.
- Modem — Links the new modem device to /dev/modem.
- CD−ROM — Links the CD−ROM device to /dev/cdrom.
- Scanner — Links the new scanner device to /dev/scanner.
- Keyboard — Runs the kbdconfig command to reconfigure the keyboard. Also, if you are using a serial console, it makes sure /etc/inittab and /etc/securetty are configured to be used by a serial console.

The following is a list of actions kudzu takes when a device is removed:

- Network — Removes the alias for the Ethernet interface (eth0, eth1, etc.).
- SCSI — Removes the alias for the SCSI host adapter (scsi_hostadapter).
- Mouse — Removes the link to /dev/mouse.
- Modem — Removes the link to /dev/modem.
- CD−ROM — Removes the link to /dev/cdrom.
- Scanner — Removes the link to /dev/scanner.

The only known problems with kudzu have to do with probing serial devices and video cards on a running Red Hat Linux system. If serial devices or older video cards are in use while kudzu is probing them, activity on those devices can be disturbed.

Monitoring System Performance

If your Linux system is being used as a multiuser computer, sharing the processing power of that computer can be a major issue. Likewise, any time you can stop a runaway process or reduce the overhead of an unnecessary program running, your Linux server can do a better job serving files, Web pages, or e−mail to the people that rely on it.

Utilities are included with Linux that can help you monitor the performance of your Linux system. The kinds of features you want to monitor in Linux include CPU usage, memory usage (RAM and swap space), and overall load on the system. The following sections describe tools for monitoring Linux.

Checking system load average with xload

One way of keeping an eye on general system performance is to open an xload window on your desktop and put it off in the corner somewhere. The xload window graphically represents the performance of your system. It periodically checks the load on the system and then charts demand on the system over time.

By default, xload updates the display every 10 seconds. Scale lines on the window help you monitor when xload has exceeded certain limits. The label in the xload window shows the system name. By running xload windows from various host computers and displaying them on your screen, you can monitor a whole set of computers at once.

Monitoring CPU usage with top and gtop

Start the top utility in a terminal window, and it displays the top CPU consuming processes on your computer. Every five seconds, top will determine which processes are consuming the most CPU time and display them in descending order on your screen.

By adding the −S option to top, you can have the display show you the cumulative CPU time that the process, as well as any child processes that may already have exited, has spent. If you want to change how often the screen is updated, you can add the −d secs option, where secs is replaced by the number of seconds between updates.

Checking virtual memory and CPU usage with vmstat

Nothing can slow down system performance more than running out of virtual memory. Waiting for CPU time can also keep processes from running efficiently. The vmstat command displays a variety of statistics that can tell you how efficiently your Linux system is running.

The following is some of the information that can be interpreted from this output:

- Where there are not a lot of processes running on this CPU, under the procs runtime column (r), you can see that in each case there were processes waiting for runtime (in the third update, nine processes were waiting to run). Up to 14 processes (b) were waiting in an uninterruptible sleep.
- Under the memory free column, you can see that the amount of idle memory was down in the third and fourth update, then recovered for the fifth update.
- Despite the demands on the CPU time, no memory was swapped to or from the hard disk (swap si and so).
- While there were processes waiting for CPU time, the processor itself was idle half the time or more in each instance (cpu id was between 50 and 97 percent). Demand from user processes (us) was greater than demand from system processes (sy).

Setting Up and Supporting Users

Overview

One of the more fundamental tasks of administering a Red Hat Linux server is setting up and supporting user accounts. Computers, after all, are tools to be used by people. Apocalyptic science fiction plots aside, without users, computers have no purpose.

When you install Red Hat Linux, you are required to create at least two user accounts: one for the root user (administrator) and one for any name you choose (regular user). Several other administrative user accounts are set up automatically that you will probably never use.

Creating User Accounts

Every person who uses your Red Hat Linux system should have a separate user account. Having a user account provides each person with an area in which to securely store files, as well as a means of tailoring his or her user interface (GUI, path, environment variables, and so on) to suit the way that he or she uses the computer.

You can add user accounts to your Red Hat Linux system in several different ways. This chapter describes how to use the useradd command to add user accounts to Red Hat Linux.

The most straightforward method for creating a new user from the shell is with the useradd command. After opening a Terminal window with root permission, you simply invoke the useradd command at the command prompt, passing the details of the new account as parameters. The only required parameter is the login name of the user, but you will probably want to include some additional information.

Option	Description
-c comment	Provide a description of the new user account. Usually just the person's full name. Replace comment with the name of the user account.
-d home_dir	Set the home directory to use for the account. The default is to name it the same as the login name and to place it in /home. Replace home_dir with the directory name to use.
-D	Rather than create a new account, save the supplied information as the new default settings for any new accounts that are created.
-e expire_date	Assign the expiration date for the account in MM/DD/YYYY format. Replace expire_date with the expiration date to use.
-f inactivity	Set the number of days after a password expires until the account is permanently disabled. Setting this to 0 disables the account immediately after the password has expired. Setting it to -1 disables the option, which is the default behavior. Replace inactivity with the number to use.
-g group	Set the primary group (as listed in the /etc/group file) that the newuser will be in. Replace group with the group name to use.
-G grouplist	Add the new user to the supplied comma-separated list of groups.
-M	Do not create the new user's home directory, even if the default behavior is set to create it.
-m	Automatically create the user's home directory and copy the files in the skeleton directory (/etc/skel) to it.
-r	Allows you to create a new account with a user ID in the range reserved for system accounts.
-s shell	Specify the command shell to use for this account. Replace shell with the command shell.
-u user_id	Specify the user ID number for the account. The default behavior is to automatically assign the next available number. Replace user_id with the ID number.

As an example, let's create an account for a new user named Mary Smith with a login name of mary. First, log in as root, then type the following command:

```
# useradd -c "Mary Smith" mary
```

Next, set Mary's initial password using the passwd command. It prompts you to type the password twice.

```
# passwd mary
New UNIX password: *******
Retype new UNIX password: *******
```

In creating the account for Mary, the useradd command performs several actions:

- Reads the /etc/login.defs file to learn the default values it should use when creating accounts.
- Parses the command line parameters to find out which default values should be overridden.
- Creates a new user entry in the /etc/passwd and /etc/shadow files based on the default values and command-line parameters.
- Creates any new group entries in the /etc/group file.
- Creates a home directory based on the user's name and located within the /home parent directory.
- Copies any files located within the /etc/skel directory to the new home directory. This usually includes login and application startup scripts.

Example

```
# useradd -m -g users -G wheel,sales -s /bin/tcsh -c "Mary Smith" mary
```

This results in a line similar to the following being added to the /etc/passwd file:

```
mary:x:500:100:Mary Smith:/home/mary:/bin/tcsh
```

In the /etc/passwd file, each line represents a single user account record. Each field is separated from the next by a colon (:) character. The field's position determines what it is. As you can see, the login name is first. The password field contains an x because we are using a shadow password file to store encrypted password data. The user ID selected by the useradd command was 500. The primary group ID is 100, which corresponds to the users group in the /etc/group file. The comment field was correctly set to Mary Smith, the home directory was automatically assigned as /home/mary, and the command shell was assigned as /bin/tcsh, exactly as specified with the useradd options.

The /etc/group file holds information about the different groups on your Red Hat Linux system and the users that belong to them. Groups are useful for allowing multiple people to share access to the same files while denying access to others. If you peek at the /etc/group file, you should find something similar to this:

```
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root,joe,mary
mail::12:mail
news::13:news
uucp::14:uucp
```

```
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
sales::500:bob,jane,joe,mary
```

Each line in the group file contains the name of a group, the group ID number associated with it, and a list of users in that group. Note that mary was added to the wheel and sales groups. Though Mary's primary group is users, her name doesn't need to be added there. The useradd command did that for you when you included the -G flag.

Setting User Defaults

The useradd command determines the default values for new accounts by reading the /etc/login.defs file. You can modify those defaults by either editing that file manually with a standard text editor or by running the useradd command with the -D option. If you choose to edit the file manually, here is what you will face:

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

MAIL_DIR /var/spool/mail

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

PASS_MAX_DAYS 99999
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7

#
# Min/max values for automatic uid selection in useradd
#

UID_MIN 500
UID_MAX 60000

#
# Min/max values for automatic gid selection in groupadd
#

GID_MIN 500
GID_MAX 60000

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

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

CREATE_HOME yes
```

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

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

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

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

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

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

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

Modifying accounts

A user needs more done to an account than just a resetting of the password. A person may become married and need the full name in the comment field changed. You may need to change the groups that user is in, or the drive that a home directory resides on. The usermod command is the tool for these tasks.

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

usermod Options for Changing Existing Accounts

Option	Description
-c comment	Change the description field of the account. You can also use the chfn command for this. Replace comment with a name or other description of the user account, placing multiple words in quotes.
-d home_dir	Change the home directory of the account to the specified new location. If the -m option is included, copy the contents of the home directory as well. Replace home_dir with the full path to the new directory.
-D	Rather than create a new account, save the supplied information as the new default settings for any new accounts that are created.
-e expire_date	Assign a new expiration date for the account, replacing expire_date with a date in MM/DD/YYYY format.
-f inactivity	Set the number of days after a password expires until the account is permanently disabled. Setting inactivity to 0 disables the account immediately after the password has expired. Setting it to -1 disables the option, which is the default behavior.
-g group	Change the primary group (as listed in the /etc/group file) that the user is in. Replace group with the name of the new group.
-G grouplist	Set the list of groups that user belongs to. Replace grouplist with a list of groups.
-l login_name	Change the login name of the account to the name supplied after the -l option. Replace login_name with the new name. This automatically change the name of the home directory; use the -d and -m options for that.
-s shell	Specify the command shell to use for this account. Replace shell with the command shell.
-u user_id	Change the user ID number for the account. Replace user_id with the new user ID number. Unless the -o option is used, the ID number must not be in use by another account.
-m	This option is used only in conjunction with the -d option. It causes the contents of the user's home directory to be copied to the new directory.

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

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

Deleting User Accounts

It is necessary to remove a user account from your Red Hat Linux system. This can be done with the userdel command. The userdel command takes a single argument, which is the login name of the account to delete. If you supply the optional -r option, it also deletes the user's home directory and all the files in it. To delete the user account with login name mary, you would type this:

```
# userdel mary
```

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

```
# userdel -r mary
```

Resetting a user's password

A common (if not the most common) problem that your users will encounter is the inability to log in. The most common causes for this are:

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

If the Caps Lock key is not on, then you probably need to reset the individual's password. Looking up the password and telling it to the user is not an option. Red Hat Linux stores passwords in an encrypted format. Instead, use the passwd command to assign a new password to the user's account. Tell the user what that new password is (preferably in person), but then set the password to expire soon so that he or she must choose one (hopefully, a new one that is more easily remembered).

If you must reset a user's password, do so with the passwd command. While logged in as root, type passwd followed by the login name you are resetting. You are prompted to enter the password twice.

```
# passwd mary
```

Creating Group

To create new group, use a groupadd command.

```
groupadd [-g gid [-o]] [-r] [-f] group
```

The groupadd command creates a new group account using the values specified on the command line and the default values from the system. The new group will be entered into the system files as needed. The option which apply to the groupadd command are

-g <gid>	The numerical value of the group id. This value must be unique, unless the –o option is used. The value must be non-negative. The default is to use the smallest ID value greater than 500 and greater than every other group. Values between 0 and 499 are typically reserved for system accounts.
-r	This flag instructs groupadd to add a system account. The first available gid lower than 499 will be automatically selected unless then –g option is also given on the command line. This is an oiption added by Red Hat.
-f	This is the gorce flag. This will cause groupadd to exit with an error when the group about to be added already exists on the system. If that is the case, the group won't be altered (or added again).

Modify Groups

groupmod command is use for modifying groups.

```
groupmod [-g gid [-o]] [-n group_name] group
```

The groupmod command modifies the system account files to reflect the changes that are specified on the comman line. The options which apply to the groupmod command are

-g <gid>	The numerical value of the group ID. This value must be unique, unless the –o option is used. The value must be non-negative. Values between 0 to 99 are typically reserved for system groups. Any files which the old group ID is the file group id must have the file group ID changed manually.
-n <group_name>	The name of the group will be changed from group to group_name.

Delete Groups

groupdel is use for deleting group.

```
groupdel group
```

The groupdel command modifies the system account files, deleting all entries that refer to group. The named group must exist. You must manually check all filesystems to insure that no files remain with the name group as the file group ID.

Using a shadow password file

In early versions of UNIX, all user account and password information was stored in a file that all users could read (although only root could write to it). This was generally not a problem because the password information was encrypted. The password was encrypted using a trapdoor algorithm, meaning the nonencoded password could be encoded into a scrambled string of characters, but that scrambled string could not be translated back to the nonencoded password.

Checking for the shadow password file

The password file is named passwd and can be found in the /etc directory. The shadow password file is named shadow and is also located in /etc. If your /etc/shadow file is missing, then it is likely that your Linux system is storing the password information in the /etc/passwd file instead. You can verify this by printing the file to the screen using the more command.

```
# more /etc/passwd
```

Something similar to the following should be displayed:

```
root:DkkS6Uke799fQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
```

```
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/u/ftp:
nobody:*:99:99:Nobody:/:
postgres:!:100:101:PostreSQL Server:/var/lib/pgsql:/bin/bash
mary:KpRUp2ozmY5TA:500:100:Mary Smith:/home/mary:/bin/sh
joe:0sXrzvKnQaksI:501:100:Joe Johnson:/home/joe:/bin/sh
jane:ptNoiueYEjwX.:502:100:Jane Anderson:/home/jane:/bin/sh
bob:Ju2vY7A0X6Kzw:503:100:Bob Renolds:/home/bob:/bin/sh
```

Backing Up and Restoring Files

The following sections describe different backup methods.

Full backup

A full backup is one that stores every file on a particular disk or partition. If that disk should ever crash, you can rebuild your system by restoring the entire backup to a new disk. Whatever backup strategy you decide on, some sort of full backup should be part of it. You may perform full backups every night or perhaps only once every week; it depends on how often you add or modify files on your system, as well as the capacity of your backup equipment.

Incremental backup

An incremental backup is one that contains only those files that have been added or modified since the last time a more complete backup was made. You may choose to do incremental backups to conserve your backup media. Incremental backups also take less time to complete. This can be important when systems are in high use during the work week and running a full backup would degrade system performance. Full backups can be reserved for the weekend when the system is not in use.

Disk mirroring

Full and incremental backups can take time to restore, and sometimes you just can't afford that downtime. By duplicating your operating system and data on an additional hard drive, you can greatly increase the speed with which you can recover from a server crash. With disk mirroring, it is usually common for the system to continuously update the duplicate drive with the most current information. In fact, with a type of mirroring called RAID 1, the duplicate drive is written to at the same time as the original, and if the main drive fails, the duplicate can immediately take over. This is called fault−tolerant behavior, which is a must if you are running a mission−critical server of some kind.

Network backup

All of the preceding backup strategies can be performed over a network. This is good because you can share a single backup device with many computers on a network. This is much cheaper and more convenient than installing a tape drive or other backup device in every system on your network. If you have many computers, however, your backup device will require a lot of capacity. In such a case, you may consider a mechanical tape loader or writable CD jukebox.

It is even possible to do a form of disk mirroring over the network. For example, a Web server may store a duplicate copy of its data on another server. If the first server crashes, a simple TCP/IP hostname change can redirect the Web traffic to the second server. When the original server is rebuilt, it can recover all of its data from the backup server and be back in business.

Selecting a Backup Medium

Armed with a backup strategy in mind, it is time to select a backup medium. Several types of backup hardware and media are available for use with Red Hat Linux. Each type has its advantages and disadvantages. The type of medium to choose depends largely on the amount of data you need to archive, how long you will store backups, how often you expect to recover data from your backups, and how much you can afford to spend.

sBackup Medium	Advantage	Advantage
Magnetic tape	High capacity, low cost for archiving massive mounts of data.	Sequential access medium, so recovery of individual files can be slow.
Writable CDs	Random access medium, so recovery of individual files is easier. Backups can be restored from any CD−ROM.	Limited storage space (approximately 650MB per CD).
Additional hard drive	Allows faster and more frequent backups. Fast recovery from crashes. No media to load. Data can be located and recovered more quickly. You can configure the second disk to be a virtual clone of the first disk, so that you can boot off of the second disk if the first disk crashes.	Data cannot be stored offsite, thus there is risk of data loss if the entire server is destroyed. This method is not well suited to keeping historical archives of the many revisions of your files. The hard drive will eventually fill up.

Working with the Desktop

Add a graphical user interface (GUI) to an otherwise unintuitive operating system, and it immediately becomes something anyone can use. Icons can represent programs and files. Linux uses the X Window System (also referred to as X11 or just X) as the framework for its graphical desktop. On top of this framework, Red Hat Linux lets you choose either (or both) of two powerful desktop environments: GNOME and KDE.

Configuring Your Desktop

If you installed Red Hat Linux as a desktop system and everything went smoothly, you should have configured your video card and chosen a desktop environment (GNOME or KDE). If so, you can skip ahead to the Starting the X Desktop section. If you were unable to configure your desktop or if you need to change it (for example, you may have added a video card or changed your monitor), this section is here for your reference.

Running Xconfigurator

The Xconfigurator command can be used to set up the links and configuration files needed to run your graphical X desktop environment. It checks that the correct X server is installed and configures the /etc/X11/XF86Config file. The following is an example of an Xconfigurator session:

1. Type Xconfigurator from the shell prompt (as the root user). A welcome screen appears.
2. Highlight the Ok button (using tab and arrow keys) and press Enter. Xconfigurator probes for your video card. If it finds one, it displays the name of the card, the X server, and the Xfree4 driver.
3. Highlight Ok and press Enter to continue. A list of monitors appears.
4. Select the type of monitor you are using. You can use Page Up and Page Down to search through the list. Type a letter to go directly to a monitor name that begins with that letter. Highlight the correct monitor, highlight Ok, and press Enter. The next window asks you to identify your video memory.

Note If your monitor is not on the list, select Custom. To install a custom monitor, you need to determine the vertical refresh rate and the horizontal sync rate to properly configure your monitor. Check the manual that comes with your monitor for that information.

5. Select how much memory is on your video card, highlight OK, and press Enter. The next window asks you to identify your clock configuration.
6. Select No Clockchip Setting (unless you know that your card requires a special clockchip setting). Highlight Ok and press Enter. The next window displays video modes that are available to your video card.
7. Select the number of colors and video resolutions for each number of colors. A higher number of colors (8−bit, 16−bit, or 24−bit) allow better quality graphics, but can slow performance. Higher resolutions (800 @@ts 600, 1024 @@ts 768, or 1152 @@ts 864) allow more space for windows, but everything is smaller. You can select several resolutions for each set of colors. Position the cursor over a selection and press the space bar to select it. Highlight Ok and press Enter to continue. The window warns you that it is about to test X.
8. Highlight Ok and press Enter. If X is working properly, you should see a pop−up window that asks if you can see it. If you don't see this window, it will timeout after a few seconds and return to your Xconfigurator. You'll need to try configuring the card again. If you do see the pop−up window, click the Yes button. You are asked if you want to start X at boot−time.
9. Click Yes (to have X start when you boot your computer) or No (to start from a text−based prompt and start X lster manually). A pop−up window alerts you that the new configuration has been saved.

Click OK and you are done.

You should now be able to start your X environment.

Understanding the XF86Config file

The XF86Config file (located in the /etc/X11 directory) contains definitions used by the X server to use your video card, keyboard, mouse, and monitor. In general, novice users should not edit this file directly, but rather,use Xconfigurator to change its contents. For some video cards, however, there is a need for some manual onfiguration to get the card working properly.

The following is a description of the basic information contained in the XF86Config file:

- Files section — Sets the locations of the RGB (color) and fonts databases.
- Server flags section — Allows you to disable abort and mode switching key sequences (usually you ill leave this section alone).
- Keyboard section — Sets keyboard settings, including the layout of the keyboard and how certain ey sequences are mapped to the keyboard.
- Pointer section — Selects the pointer you are using (typically a mouse linked to /dev/mouse). Also sets speed and button emulation, when appropriate.
- Monitor section — Sets the type of monitor, along with its horizontal sync rate, vertical refresh rate, and settings needed to operate at different resolutions.
- Screen section — Binds together the graphics board and monitor information to be referenced later by the ServerLayout section.
- Graphics device section — Identifies your video card and, optionally, video RAM and clock information for the chipset.
- ServerLayout section — Sets server definitions for different X servers (if necessary). For further information on the XF86Config file, see the XF86Config man pages (type man XF86Config).

Starting the X Desktop

There are several different ways you can start your desktop in Red Hat Linux. If Red Hat Linux starts up and you see a graphical login screen, you can just log in and your desktop environment should appear. If Red Hat Linux starts up to a simple text−based login prompt, you can have the desktop environment start after you log in (either manually or automatically). Each of these methods is described in this section.

- At boot−time — After your video card and monitor are properly configured, you may want to have the X GUI start automatically when Red Hat Linux is booted. You can do this by setting your system initialization state to 5. After you log in, you will see the GNOME or KDE desktop environments
- From the shell (or starting it yourself) — If you have logged in to a shell interface, you can start the GUI at any time using the startx command. When you quit from the GUI, you will be back at your shell prompt.
- At login time — To start with a text−based login screen, but have the X GUI started after you log in, you can add the startx command to one of your personal startup files (such as $HOME/.bash_profile). Each of these methods results in the startup of the X GUI and either the GNOME or KDE desktop environment. Along with the GNOME desktop environment, the sawfish window manager (or possibly some other window manager) starts up in order to provide the look−and−feel of the desktop.

Starting the GUI at boot time

After Red Hat Linux boots up, a Red Hat logo and a GNOME (default) or KDE login window appears. You are ready to start using Red Hat Linux from an X Window GUI (probably GNOME and sawfish).

Type your login and password, as prompted, and your personal desktop is displayed. It is possible to change several important options when you log in using the graphical login screen. In fact, you can even select a desktop environment other than the default. Here are menu options you can select:

- Session — From this menu item, you can select the desktop interface that is used with your login session. Besides GNOME, you can possibly choose Default, Failsafe, WindowMaker, or KDE as your interface (depending on which interfaces you have installed).
- Language — You can choose from more than a dozen languages to use during your desktop session (all languages are not completely supported).
- System — Instead of logging in, you can choose Reboot or Halt from the System selection to restart or halt Red Hat Linux. The graphical login screen is started by the xdm command (which stands for X Display Manager). The xdm process manages both logging in and starting the GUI for your console monitor, as well as graphical logins from other computers and X terminals. Normally, you start xdm by

setting the system's default run state to 5. You can determine your system's default run state by checking the /etc/inittab file for an entry that looks similar to the following:

```
id:5:initdefault:
```

If the initdefault state is 3, the system boots to a text−based login prompt. At the bottom of that file, you will see that the xdm command is run in system state 5. (Instead of xdm, a GNOME version of xdm called prefdm or gem may run. Either command results in the same functions as xdm.) The identity of many configuration files is contained in the xdm−config file (usually in the /etc/X11/xdm directory). Here are some of the files identified in xdm−config that may interest you.

- Error Log (/var/log/xdm−error.log) — Contains error messages output from xdm. You can check this log if you have problems starting X.
- Servers (/etc/X11/xdm/Xservers) — Identifies the X server used for your display. If your computer has extra displays or X terminals that access it, you can add X server entries here.
- Login Setup (/etc/X11/xdm/Xsetup_0) — Identifies the client processes that are run on the console display login screen. These processes include the xsetroot command (to set the background color) and the xsri command (to display the Red Hat logo).
- Session Setup (/etc/X11/xdm/Xsession) — Defines how to start up your window manager session.

Note: Settings that you put into any of the above files, such as the Xsession file, are likely to be overridden by settings that are specific to your desktop environment (GNOME or KDE). For example, I changed the background color of my desktop in the Xsession file to red (xsetroot −solid 'red'). When X started, the screen background flashed red; then GNOME took over and changed the background to blue.

Using the GNOME Desktop Environment

GNOME (pronounced guh-nome) provides the desktop environment that you get by default when you install Red Hat Linux. This desktop environment provides the software that is between your X Window System framework and the look-and-feel provided by the window manager. GNOME offers a stable and reliable desktop environment, with a few cool features built in.

GNOME is not a window manager, so it must be used with a window manager to provide such things as window borders and window controls. Currently, sawfish is the default window manager with GNOME. You can, however, use other window managers with GNOME, including:

- Enlightenment
- Window Maker
- Twm

You can change to any of these window managers using the GNOME Window Manager window, which you can start from the GNOME menu by choosing Programs → Settings → Desktop → Window Manager. (How to configure GNOME is described later in this section.)

This section describes the GNOME desktop environment and ways of using it. If your Red Hat Linux system is configured to use a GUI by default, you simply need to log in from the graphical login screen. Otherwise, type **startx** from a shell prompt after you log in.

The GNOME desktop that appears the first time it is started by a new user account includes the GNOME panel, desktop area, desktop icons, and file manager. Descriptions of those elements follow. The panel contains most of the controls you need to use the desktop. When it first starts, the panel contains buttons for starting applications, a list of active applications, several applets, and a few controls. This is what you should see:

- **Main menu button** (footprint icon) — Click this button to see a list of menu items from which you can select. These menu items include selections for starting applications (such as the Run program window, Lock screen, and Log out selections) and submenus of selections. The Programs menu (under the main menu) includes the following submenus: Applications, Utilities,Games, Graphics, Internet, Multimedia, Settings, and System. There is also a Help System selection from this menu.

- **Lock screen** (lock icon) — Lets you lock the screen so it can be reopened using your password only.
- **Start Here** (mushroom icon) — Opens a Start Here window, which is actually a file manager window displaying icons for configuring GNOME.
- **Terminal window** (terminal icon) — Opens a Terminal window, which provides access to a Red Hat Linux shell.
- **Mozilla browser** (red monster head logo) — Opens the Mozilla browser window. This software is for browsing the World Wide Web or for running related programs to use e-mail, to compose Web pages, or to view newsgroups. (Mozilla has recently replaced Netscape Navigator as the default browser for Red Hat Linux, though Netscape is still available from the GNOME menu.)
- **Taskbar** — Shows the tasks that are currently running on the desktop. The window that is currently active appears pressed in. Click a task to toggle between opening and minimizing the window.
- **Pager** — This applet shows a tiny view of the multiple desktop areas of your GNOME environment. By default, there are four desktops available. You can change to a different desktop by clicking the area in the pager. Each minidesktop area shows small representations of the active windows within that desk. Click any active application to make it the current application. A sticky application is one that sticks to the same place on the screen, regardless of which desktop you move to. People stick things such as clocks or e-mail notification windows.
- **Clock** — This applet shows the current date and time.

Other useful GNOME controls are located in the following places:

- **Desktop Area** — This is the screen area on which you can use the windows, icons, and menus that make up the GNOME desktop. Right-click the desktop area to see a menu of options.
- **File Manager** — This window contains a graphical means of moving up and down your Red Hat Linux file system, opening files, and running applications. It supports expected features for creating, opening, copying, deleting, and moving files and folders. However, it also has some special features, such as drag-and-drop, launching applications based on MIME type, and file finding. The File Manager is called Nautilus. Nautilus is now the default file manager for Red Hat Linux 7.2.
- **Desktop icons** — Icons on the desktop each represent a directory, application, or file that you can work with. You can double-click a desktop icon to open it, drag-and-drop it to move it to another location, or right-click it to see a menu of options related to the icon. The icons added to your Red Hat Linux GNOME desktop include one that opens a File Manager to your home directory, one that opens the Start Here window, and a Trash icon for deleting files. The GNOME desktop is quite intuitive. However, you may want to read some of the following descriptions of GNOME if you are a novice user or if you want to learn some of the less obvious features of the GNOME desktop.

Using the GNOME panel

The GNOME panel is intended to be the place from which you manage your desktop. From this panel you can start applications (from buttons or menus), see what programs are active, and monitor how your system is running. There are also many ways to change the panel, such as by adding applications or monitors, or changing the placement or behavior of the panel. Click the GNOME menu, then select the Panel menu. From this menu, you can perform a variety of functions, including:

- **Add to panel** — You can add an applet, menu, launcher, drawer, button (logout, lock, or run), a swallowed application, or a status dock.
- **Create panel** — You can create additional panels for your desktops in different styles.
- **Remove this panel** — You can delete the current panel. Procedures for each of these functions are described in the following sections. For information on changing properties of the panel itself, see the section on changing GNOME preferences.

Adding an applet

There are several small GNOME applications, called applets, that you can run directly on the GNOME panel. These applets can show information you may want to see on an ongoing basis or may just provide some amusement. To see what applets are available and to add applets that you want to your panel, perform the following steps:

1. Click the GNOME menu button on the panel (the paw print icon), then select Panel → Add to panel → Applet.
2. Select an applet from one of the following categories:
    - **Amusements** — Includes a puzzle game (Fifteen), eyes that follow your mouse around, and a few other amusements.

- **Monitors** — Includes useful tools for monitoring CPU/Memory usage, CPU load, disk usage, load average, memory load, network load, and swap load.
- **Multimedia** — Includes applets for playing CDs or MP3 audio files or using an audio mixer.
- **Network** — Includes applets for checking for AOL instant messaging (GAIM), checking incoming mail, displaying stock prices or headlines, showing modem incoming and outgoing traffic, dialing a PPP connection, or opening a Web site (HTML page).
- **Utility** — Includes applets for monitoring battery levels, mounting floppy drives, managing print queues, and performing other useful tasks.
- **Clock** — Runs a clock applets.

3. Select the applet to add from the category you selected. The applet appears on the panel, ready for you to use. After an applet is installed, right-click it to see what options are available. If you don't like its location, right-click it, click Move applet, slide the mouse until the applet is where you want it, then click again to set its location. If you no longer want an applet to appear on the panel, right-click it, and then click Remove from panel. The icon representing the applet will disappear. If you find that you have run out of room on your panel, you can add a new panel to another part of the screen, as described in the next section.

Adding another panel

You can have several panels on your GNOME desktop. You can either add edge panels (that run along the entire bottom, top, or side of the screen) or a corner panel (that only expands as needed to show the applets it contains). To add a panel, do the following:

1. Click the GNOME menu button on the panel (the paw print icon), then select Panel → Create panel.
2. Select from the following menu items:
   - **Menu panel** — Places a menu bar across the top of the screen.
   - **Edge panel** — Creates a panel along the edge of the screen (top, sides, or bottom).
   - **Aligned panel** — Adds a panel that stretches out only as far as is required to display the applets it contains.
   - **Sliding panel** — Creates a panel that stretches out only as far as required to display the applets it contains. When you hide/unhide the panel, it closes in the middle of the display instead of just at the edges.
   - **Floating panel** — Adds a panel that begins from the middle of the edge of a display (instead of from a corner).

After you've added a panel, you can add applets or application launchers to it as you did to the default panel. You can change an edge panel to a corner panel (or vice versa), by right-clicking the edge panel, and then selecting Panel → Properties → Type → and selecting the type of panel that you want. To remove a panel, right-click it and select Panel → Remove this panel.

Adding an application launcher

Icons on your panel that show the terminal and the Mozilla logo are used to launch Terminal and Netscape Communicator applications, respectively. You can add your own icons to launch applications from the panel. Icons that are available to use to represent your application are contained in the /usr/share/pixmaps directory. These icons are either in png or xpm formats. If there isn't an icon in that directory that you want to use, create your own and assign it to the application. To add a new application launcher to the panel, do the following:

1. Right-click the panel.
2. Select Panel → Add to panel → Launcher. The Create launcher applet window appears.
3. Provide the following information for the application that you want to add:
   - **Name** — A name to identify the application.
   - **Comment** — A comment describing the application. This information appears when you later move your mouse over the launcher.
   - **Command** — The command line that is run when the application is launched. You should use the full path name, plus any required options.
   - **Type** — Select either Application (if you are launching an application), URL (to open a Web address in a browser) or Directory (if you are opening a directory in a File Manager window).
   - **Run in Terminal** — If it is a character-based or ncurses application. (Applications written using the curses library run in a Terminal window but offer screen-oriented mouse and keyboard controls.)
4. Click the Icon box (it may say No Icon). Select one of the icons shown and click OK. Alternatively, you can browse the Red Hat Linux file system to choose an icon. (Icons should be in png or xpm graphics formats.)

5. Click OK.

The application should now appear in the panel. Click it to start the application.

Changing panel properties

There are both global and specific panel properties that you may want to consider changing. To change global panel preferences, open the GNOME menu and choose Panel → Global Preferences. The Panel window appears. Five tabs are used to change the behavior of your GNOME panels. The Animation tab enables you to change animation speeds that affect how windows and icons are moved and minimized. The Buttons tab lets you set tiles used for the backgrounds of Launcher, Drawer, Menu, and Special buttons. The Panel Objects tab is used to set how applets change positions on the panel when you move them. The Menu tab sets the positions of different global menu items on the GNOME menu. The Miscellaneous tab enables you to assign properties that are used with tool tips and various other miscellaneous options. To open the Panel properties window that applies to a specific panel, open the GNOME menu on that panel. Then choose Panel → Properties → All Properties. The Panel properties window that appears enables you to change the following values:

- **Type** — You can see whether the panel is an Edge, Aligned, Sliding, or Floating panel.
- **Hiding policy** — You can select whether or not a panel is explicitly hidden (by clicking the hide/unhide buttons) or automatically hidden (appearing only when the mouse pointer is in the area).
- **Hide buttons** — You can choose whether or not the Hide/Unhide buttons (with pixmap arrows on them) appear on the edges of the panel.
- **Size and position** — You can select the size of your panel, from Ultra Tiny (12 pixels) to Ridiculous (128 pixels).
- **Background** — You can assign a color to the background of the panel, assign a pixmap image, or just leave the standard background (sort of gray). Click the Background tab, and then select the pixmap (from the /usr/share/pixmaps or other directory) or color to assign to the panel. Tip I usually turn on the AutoHide feature and turn off the Hide buttons. Using AutoHide gives you more space to work with on your desktop. When you move your mouse to the edge where the panel is, it pop up — obviating any need for the Hide buttons.

Using the Nautilus file manager

At one time, file managers did little more than let you run applications, create data files, and open folders. These days, as the information a user needs expands beyond the local system, file managers are expected to also display Web pages, access FTP sites, and play multimedia content. The Nautilus file manager, which is the default GNOME file manager, is an example of just such a file manager. When you open the Nautilus file manager window (from the GNOME main menu or by launching a directory), you will see a sidebar in the left column and the files and directories contained in the current directory in the right column. Figure 4-3 is an example of the File manager window displaying the home directory of a user named jake (/home/jake). Icons on the toolbar of the Nautilus window let you move forward and back among the directories and Web sites you visit. To move up the directory structure, click the Up arrow. To refresh the view of the folder or Web page, click the Refresh button. The Home button takes you to your home page and the Web search button lets you search the Web. Many data files are represented by icons that indicate the type of data they contain. The contents or file extension of each file can determine which application is used to work with the file. Or, you can right-click an icon to open the file it represents with a particular application or viewer. Some of the more interesting features of Nautilus are described below:

- **Sidebar** — The left column of the screen consists of a sidebar. From the sidebar, you can click on tabs that represent different types of information you can select. The Tree tab shows a tree view of the directory structure, so you can easily traverse your directories. The News tab lets you select from a variety of news sites and has headlines displayed from those sites.

  The Notes tab lets you add notes that become associated with the current Directory or Web page. The Help tab opens links to the GNOME Help system. The History tab displays a history of directories and Web sites you have visited, allowing you to click those items to return to the sites they represent.
- **Web browsing** — By typing a Web address (URL) in the Location box on the Nautilus window or by clicking on a link, Nautilus acts as a Web browser by displaying the requested content. Select Web search to have your favorite search engine displayed to search for Web sites (http://www.google.com/ is used by default).
- **MIME types and file types** — To handle different types of content that may be encountered in the Nautilus window, you can set applications to respond based on MIME type and file type. With a directory being displayed,

click a file for which you want to assign an application. Click either Open With an Application or Open With a Viewer. If no application or viewer has been assigned for the file type, click OK to be able to select an application. From the File Types and Programs window, you can add an application based on the file extension and MIME type representing the file. Cross-Reference For more information in MIME types, see the description of MIME types in the "Changing GNOME preferences" section later in this chapter.

- **Drag-and-Drop** — You can use drag-and-drop within the Nautilus window, between the Nautilus and the desktop, or between multiple Nautilus windows. As other GNOME-compliant applications become available, they are expected to also support the GNOME drag-and-drop feature.

Changing the desktop

From the Control Center window, you can change the Background, Launch Feedback, Panel, Screensaver, Theme Selector, or Window Manager for your GNOME desktop. After you open the Control Center window, select one of those categories. Here are the preferences that you can change from each:

- **Background** — From the Background preferences, you can choose a single solid color, a two-color gradient (shading from top to bottom or left to right), or a pixmap image to use as wallpaper. If you choose a single solid color, click the color box under Primary Color, choose a color from the palette, and select OK. To have the color fade into another color, click Vertical or Horizontal Gradient, and then select the second color. Choosing Vertical has the colors transform from top to bottom, while Horizontal causes the colors to change from left to right. To use wallpaper for your background, click Browse. You can choose from a variety of images in the /usr/share/nautilus directory. Then, choose to have the image tiled, centered, scaled (in proportion), or scaled (using any proportion to fill the screen).
- **Launch Feedback** — From the Launch Feedback preferences, you can indicate the kind of behavior the GNOME exhibits while an application is being launched. While an application is being launched from the desktop, you can enable feedback in a tasklist, have an hourglass appear for the mouse cursor, have a splash screen displayed while the application launches, or have an animated star displayed.
- **Panel** — From the Panel section, you can change global behavior of the panels on your desktop. These selections are described earlier in the Using the GNOME Panel section.
- **Screensaver** — You can choose dozens of screensavers from the Screen Saver preferences window. Select Random Screensaver to have your screen saver chosen randomly, or select one that you like from the list. Next, choose how long your screen must be idle before the screen saver starts (default is 20 minutes). You can also choose to require a password or to enable power management to shut down your monitor after a set number of minutes.
- **Theme Selector** — You can choose to have an entire theme of elements be used on your desktop. A desktop theme affects not only the background, but also the way that many buttons and menu selections appear. There are about a dozen themes available with the Red Hat Linux distribution. See the sidebar "Finding Desktop Themes" for information on how to get additional desktop themes.

Using the KDE Desktop Environment

The KDE desktop was developed to provide an interface to Linux and other UNIX systems that could compete with MacOS or Microsoft Windows operating systems for ease of use. Integrated within KDE are tools for managing files, windows, multiple desktops, and applications. If you can work a mouse, you can learn to navigate the KDE desktop.

The lack of an integrated, standardized desktop environment in the past has held back Linux and other UNIX systems from acceptance on the desktop. While individual applications could run well, you rarely could drag-and-drop files or other items between applications. Likewise, you couldn't open a file and expect it to launch the correct application to deal with it. KDE provides a platform for developers who want to create applications that can easily share information and detect how to deal with different types of data. The following section describes how to get started with KDE. This includes using the KDE Setup wizard, maneuvering around the desktop, managing files, windows, virtual desktops, and adding application launchers.

Starting with KDE

You can select the KDE desktop from the login screen (provided that both GNOME and KDE are installed). Choose Session → KDE. Then, type your login name and password, as prompted.

KDE desktop described

When you start the KDE desktop for the first time, you will probably see the Desktop Settings Wizard window, some icons on the desktop, and the panel (bottom). Here are some quick descriptions of what you can expect to find on the desktop:

- **Desktop Settings Wizard** — This window appears when you first log in to KDE. It allows you to configure your desktop to your liking. I recommend you step through this window to configure your desktop initially. The window lets you:
  - Choose your country and language.
  - Set the system behavior to be in the style of KDE, other UNIX systems, Microsoft Windows, or Apple MacOS. Select each one to see how features would be set before choosing which one to use.
  - Turn on more or fewer special effects. More effects may look nicer, but be harmful to system performance. If you have a slow processor, you probably want fewer special effects.
  - Choose a theme to use for your desktop. The theme effects the look of your screen, panel, and icons, as well as the location of some controls.
  - Change the content that is added to the KDE panel.
- **Desktop icons** — The desktop starts with several icons on it to provide quick access to selected features. By default, you will probably have icons that give you access to your Home directory, CD-ROM, and floppy disk drive. There will also be icons representing the KDE Control Panel, the Red Hat Web site, the Trash can, and Linux Documentation.
- **Panel** — Provides some quick tools for launching applications and managing the desktop. You can adapt the panel to your needs by resizing it, adding tools, and changing the look and feel.
- **K Menu button** — This panel button has the letter *K* on it. Click this button to see a menu of applications, utilities, and configuration tools that are available to run on your KDE desktop.
- **Show Desktop** — This button toggles on and off the elements on your desktop, so you can see the full desktop background. This makes it easier to see your menus and icons, so you can easily start a new application when your desktop is cluttered.
- **Terminal (shell)** — This button resembles a computer terminal with a shell in front of it. Click this button to open a Terminal window. The Terminal window provides a way for you to enter commands at a shell prompt.
- **KDE Control Center** — This button looks like a computer screen, with half of the screen covered by a remote control box. Click it to open the KDE Control Center window. This window is your primary tool for configuring your KDE desktop. (This interface is described in detail a little later.)
- **KDE Help** — This button is an image of a lifesaver. Click it to open the KDE Help window. From this window, you can browse lists of KDE applications, Linux man page listings, or info command contents. You can also do a keyword search to find keywords in the different online documents that are available.
- **Home Directory** — This panel button looks like a house in front of a file folder. Click this button to open your home folder in a file manager window.
- **Konqueror Web Browser** — This panel button looks like a globe with pointy things on it. Click it to open the Konqueror Web Browser. Konqueror can be used to traverse your computer's file system as well as the Web.
- **Kmail** — This panel button is the image of an E with a letter resting on it. Click it to open the Kmail window for sending, receiving, and managing your e-mail.
- **Virtual Desktops** — There are four virtual desktops available to you, by default. These are labeled 1, 2, 3, and 4. You begin your KDE session on virtual desktop 1. You can change to any of the four desktops by clicking it.
- **Window List button** — This is a slender button with an up-arrow on it. Click it to see a list of virtual desktops and open windows on your desktop. Click an entry to go right to the virtual desktop or window you select.
- **Taskbar** — Shows the tasks that are currently running on the desktop. The window that is currently active appears pressed in. Click a task to toggle between opening and minimizing the window.
- **Lock Desktop** — This button looks like a small lock. Click it to lock your desktop from access. You need to enter your user password to access the desktop again.
- **Logout** — Click this button to end your KDE session. A window asks if you want to save the current state of your desktop to return to the next time you log in to KDE.

- **Klipper** — This button looks like a clipboard. Click it to see a list of contents on your desktop's clipboard.
- **Korganizer** — This button contains an image of a calendar. Click it to open the Korganizer window. Korganizer is an online calendar and to-do list.
- **Clock** — The current time and date are shown in the far right side of the panel. Click it to see a calendar of the current month. Click arrow keys on the calendar to step forward and back to other months.

Getting around the desktop

Navigating the desktop is done with your mouse and keyboard. You can use a two-button or three-button mouse. Using the keyboard to navigate requires using different Alt and Ctrl key sequences.

Using keystrokes

If you don't happen to have a mouse or you just like to keep your hands on the keyboard, there are several keystroke sequences you can use to navigate the desktop. Here are some examples:

- **Step through desktops** (Ctrl+Tab) — To go from one virtual desktop to the next, hold down the Ctrl key and press the Tab key until you see the desktop that you want to make current. Then release the Ctrl key to select that desktop.
- **Step through windows** (Alt+Tab) — To step through each of the windows that are running on the current desktop, hold down the Alt key and press the Tab key until you see the one you want. Then release the Alt key to select it.
- **Open Run Command box** (Alt+F2) — To open a box on the desktop that lets you type in a command and run it, hold the Alt key and press F2. Next, type the command in the box and press Enter to run it. You can also type a URL into this box to view a Web page.
- **Close the current window** (Alt+F4) — To close the current window, press Alt+F4.
- **Close another window** (Ctrl+Alt+Esc) — To close an open window on the desktop, press Ctrl+Alt+Esc. When a skull and cross bones appears as the pointer, move the pointer over the window you want to close and click the left mouse button. (This is a good technique for killing a window that has no borders or menu.)
- **Switch virtual desktops** (Ctrl+F1, F2, F3 or F4 key) — To step through virtual desktops, press and hold the Ctrl key and press F1, F2, F3, or F4 to go directly to desktop one, two, three or four, respectively. You could do this for up to eight desktops, if you have that many configured.
- **Open window operation menu** (Alt+F3) — To open the operations menu for the active window, press Alt+F3. When the menu appears, move the arrow keys to select an action (Move, Size, Minimize, Maximize, etc.), then press Enter to select it.

Managing files with the Konqueror File Manager

The Konqueror File Manager helps elevate the KDE environment from just another X window manager to an integrated desktop that can compete with GUIs from Apple Computing or Microsoft. The features in Konqueror rival those that are offered by those user-friendly desktop systems.

Some of Konqueror's greatest strengths over earlier file managers are the following:

- **Network desktop** — If your computer is connected to the Internet or a LAN, features built into Konqueror let you create links to files (using FTP) and Web pages (using HTTP) on the network and open them within the Konqueror window. Those links can appear as file icons in a Konqueror window or on the desktop. When a link is opened (single-click), the contents of the FTP site or Web page appears right in the Konqueror window.
- **Web browser interface** — The Konqueror interface works like Netscape Navigator, Mozilla, Internet Explorer, or another Web browser in the way you select files, directories, and Web content. A single-click opens a file, link to a network resource, or application program. You can also open content by typing Web-style addresses in a Location box.
- **File types and MIME types** — If you want a particular type of file to always be launched by a particular application, you can configure that file yourself. KDE already has dozens of MIME types defined that can automatically detect particular file and data types and start the right application. There are MIME types defined for audio, image, text, video, and a variety of other content types. Of course, you can also perform many standard file manager functions with Konqueror. For manipulating files, you can use features like Select, Move, Cut, Paste, and Delete. You can search directories for files, create new items (files, folders, and links, to name a few), view histories of the files and Web sites you have opened, and create bookmarks.

Working with files

Because most of the ways of working with files in Konqueror are quite intuitive (by intention), I'll just give a quick rundown of how to do basic file manipulation:

- **Open a file** — Single-click the left mouse button on a file. The file will open in the default application set for the file type. For example, a plain-text file will open in the Text Editor (kedit). You can also open a directory (to make it the current directory), application (to start the application), or link (to open the target of a link) in this way.
- **Choose an application** — Single-click the right mouse button. When you right-click a data file, select the Open With menu. The menu that appears shows which applications are set up to open the file.
- **Delete a file** — Single-click the right mouse button → Delete. You are asked if you really want to delete the file. Click Yes to permanently delete it.
- **Copy a file** — Single-click the right mouse button → Copy. This copies the file to your clipboard. After that, you can paste it to another folder. (See the "Move a file" bullet for a drag-and-drop method of copying.)
- **Paste a file** — Single-click the right mouse button (on an open area of a folder) → Paste. A copy of the file you copied previously is pasted in the current folder.
- **Move a file** — With the original folder and target folder both open on the desktop, press and hold the right mouse button on the file you want to move, drag the file to an open area of the new folder, and release the mouse button. From the menu that appears, click Move. (You could also copy or create a link to the file using this menu.)
- **Link a file** — Drag-and-drop a file from one folder to another (as described in "Move a file"). When the menu appears, click Link Here. (A linked file lets you access a file from a new location without having to make a copy of the original file. When you open the link, a pointer to the original file causes it to open.)

There are also several features for viewing information about the files and folders in your Konqueror windows:

- **View quick file information** — Position the mouse pointer over the file. When a mouse pointer is over a file in a Konqueror window, information appears in the window footer. This includes the file name, file size, and file type.
- **View hidden files** — Select View → Show Hidden Files. This allows you to see files that begin with a dot (.). Dot files tend to be used for configuration and don't generally need to be viewed in your daily work.
- **View file system tree** — Select View → View Mode → Tree View. This presents a tree view of your folder, displaying folders above the current folder in the file system. You can click a folder in the tree view to jump directly to that folder. There are also Multicolumn, Detailed List and Text views available.
- **Change icon view** — Select View → Icon Size, and then choose Large, Medium, or Small to select the size of the icons that are displayed on the window. You can also choose Default Size, to return to the default icon size.

Configuring the desktop

If you want to change the look, feel, or behavior of your KDE desktop, the best place to start is the KDE Control Center. The KDE Control Center window lets you configure dozens of attributes associated with colors, fonts, backgrounds, and screensavers used by KDE. You can also change attributes relating how you work with windows and files.

To open the KDE Control Center from the desktop, either click the KDE Control Center icon on the Panel or click the K button → Control Center. The KDE Control Center window appears. Click the plus (+) sign next to topics you want to configure. Then select the particular item you want to configure. The following sections describe some of the features you can configure from the KDE Control Center.

Changing the display

There are several ways you can change the look and feel of your desktop display. Under the Look & Feel topic (click the +), you can change Background, Colors, Desktop, Fonts, Icons, Key Bindings, Launch Feedback, Panel, Screensaver, Style, Taskbar, Theme Manager, and Window Behavior, and Window Decoration. Here are a few of the desktop features you may want to change:

- **Change the background** — Under the Look & Feel heading in the KDE Control Center, select Background. You can remove the "X" next to Common Backgrounds to choose to have a common background for all four of your virtual desktops or assign backgrounds to individual desktops.
  First select the Mode: Flat (single color); Pattern (two colors and click Setup to select a pattern); Background Program (click Setup to choose a program to run on the background); Horizontal Gradient (two colors that fade from left to right); Vertical Gradient (two colors that fade from top to bottom); Pyramid Gradient (two colors that

fade from outside in); Pipecross Gradient (two colors that fade from the outside to a cross in the middle); and Elliptic Gradient (two colors that fade from outside to an ellipse in the middle).
If you prefer to use a wallpaper on the background, click the Wallpaper tab and select a desktop. Choose a Mode (to indicate the position of the wallpaper) and click the Wallpaper box to scroll down to choose the wallpaper. If you have a JPEG image you would like to use instead, click the Browse button to select the image you want from your file system. Click the Multiple box if you want to assign several wallpapers that change at set intervals. Click Apply to apply your selections.

- **Change the screensaver** — Under the Look & Feel heading, select Screensaver. From the window that appears, select from about 25 different screen savers. Click Setup to modify the behavior of the screen saver. Under settings, select how many minutes of inactivity before the screensaver turns on. You can also click Require password to require that a password be entered before you can access your display after the screensaver has come on.

Tip If you are working in a place where you want your desktop to be secure, be sure to turn on the Require Password feature. This prevents others from gaining access to your computer when you forget to lock it or shut it off. If you have any virtual terminals open, switch to them and type vlock to lock each of them as well. (You need to install the vlock package if the vlock command isn't available.)

- **Change fonts** — You can assign different fonts to places that fonts appear on the desktop. Under the Look & Feel heading, select Fonts. Select one of the categories of fonts (General, Fixed width, Desktop icon, File manager, Toolbar, Menu, and Window title fonts). Then click the Choose box to select a font from the Select Font box that you want to assign to that category. If the font is available, you will see an example of the text in the Sample text box.

Tip If you want to use 100 dpi fonts, you need to add an entry for 100 dpi fonts to the /etc/XF86Config file. After you make that change, you need to restart the X server for it to take effect.

Other attributes you can change for the selected fonts are size (in points) and character set (to select an ISO standard character set). Select Apply to apply the changes

- **Change the colors** — Under the Look & Feel heading in the KDE Control Center, select Colors. The window that appears lets you change the color of selected items on the desktop. Select a whole color scheme from the Color Scheme box. Or select an item from the Widget color box to change a particular item. Items you can change include text, backgrounds, links, buttons, and title bars.

XDM

The graphical login screen is started by the xdm command (which stands for X Display Manager). The xdm process manages both logging in and starting the GUI for your console monitor, as well as graphical logins from other computers and X terminals.

Normally, you start xdm by setting the system's default run state to 5. You can determine your system's default run state by checking the /etc/inittab file for an entry that looks similar to the following:

```
id:5:initdefault:
```

If the initdefault state is 3, the system boots to a text−based login prompt. At the bottom of that file, you will see that the xdm command is run in system state 5. (Instead of xdm, a GNOME version of xdm called prefdm or gem may run. Either command results in the same functions as xdm.)

When xdm starts up, it reads a series of configuration files that set up both the login screen and the desktop environment that appears. The identity of many configuration files is contained in the xdm−config file (usually in the /etc/X11/xdm directory). Here are some of the files identified in xdm−config that may interest you:

- **Error Log** (/var/log/xdm−error.log) — Contains error messages output from xdm. You can check this log if you have problems starting X.
- **Servers** (/etc/X11/xdm/Xservers) — Identifies the X server used for your display. If your computer has extra displays or X terminals that access it, you can add X server entries here.
- **Login Setup** (/etc/X11/xdm/Xsetup_0) — Identifies the client processes that are run on the console display login screen. These processes include the xsetroot command (to set the background color) and the xsri command (to display the Red Hat logo).
- **Session Setup** (/etc/X11/xdm/Xsession) — Defines how to start up your window manager session.

Note: Settings that you put into any of the above files, such as the Xsession file, are likely to be overridden by settings that are specific to your desktop environment (GNOME or KDE). For example, I changed the background color of my desktop in the Xsession file to red (xsetroot −solid 'red'). When X started, the screen background flashed red; then GNOME took over and changed the background to blue.

In runlevel 5, the /etc/inittab runs a script called /etc/X11/prefdm. The prefdm script executes the preferred X display manager . gdm, kdm, or xdm, depending on the contents of the /etc/sysconfig/desktop _le.

When the system boots into runlevel 5 a special X client application, called a display manager is launched. A user must authenticate using the display manager before any desktop environments or window managers are launched. Depending on the desktop environments installed on the system, three different display managers are available to handle user authentication.

- **gdm:** The default display manager for Red Hat Linux, gdm allows the user to con_gure language settings, shutdown, restart or log in to the system.
- **kdm**: KDE's display manager which allows the user to shutdown, restart or log in to the system.
- **xdm**: A very basic display manager which only lets the user log in to the system.

Understanding Red Hat Linux and Local Area Networks

Connecting the computers in your organization via a LAN can save you a lot of time and money. The amount of money you put into networking hardware, even in a small configuration (less than five or six users), can save you from buying multiple printers, backup media, and other hardware. Add a single, shared Internet connection and you no longer need multiple modems and Internet accounts.

With a LAN, you don't have to run down the hall anymore with your file on a floppy disk to print it on your friend's printer. Information that had to wait for the mailroom to make the rounds can be sent in an instant to anyone (or everyone) on your LAN.

With a LAN, you begin to open the greatest potential of Red Hat Linux — its ability to act as a server on a network. Because Red Hat Linux is more robust and feature−rich than other computing systems (certainly for the price), adding it to your LAN can provide a focal point to workstations that could use Red Hat Linux as a file server, a mail server, a printer server, or a news server. (Those features are described later in this book.) Creating a LAN and configuring it to be useful consists of three steps:

1. **Setting up the hardware** — This entails choosing a network topology, purchasing the equipment you need, and installing it (adding cards and connecting wires).
2. **Setting up Ethernet** — Red Hat Linux must be able to recognize the Ethernet card in your computer, install a driver for it, and make it available for use by Linux. (For supported cards, this is done easily during Red Hat Linux installation.) Ethernet is the protocol that enables messages to get from one machine to another on your LAN.
3. **Configuring TCP/IP** — To use most of the networking applications and tools that come with Red Hat Linux, you must have TCP/IP configured. TCP/IP lets you communicate not only with computers on your LAN, but to any computers that you can reach on your LAN, modem, or other network connection.

Ethernet as the underlying network because it is by far the most popular LAN protocol. Protocols are rules for communications between computers. To expand beyond your LAN — for example, to share an Internet connection from your LAN to the Internet — there are several other protocols that you could use. Ethernet is a CSMA−CD type of network, which stands for *Carrier Sense Multiple Access with Collision Detection*. On this type of network, data is

broadcast on the network for all to see, then picked up by the computer for which the information is intended. The collision detection part is what helps the network detect and recover from data collisions.

Choosing peer−to−peer vs. client/server models

Although the Ethernet hub sees all computers on the network as equal, the computers on that hub can actually play different roles. The models used to describe the two general types of computing environments are client/server and peer−to−peer. Red Hat Linux is usually described as a server computer. However, Red Hat Linux can also function quite happily as a client machine or in a peer−to−peer network.

Client/Server model

In a client/server model, one or more server computers manage most network services. The server acts as a focal point for administration and security of the network. It may also control printers, databases of information, backup media, and other resources that need to be made available to client computers on the network. The client is the person (or more correctly, the software program) that requests services from the network.

A server computer can act as a server for some or all of these services:

- **Print Server** — This maintains and manages one or more printers.
- **File Server —** A central repository for documents and databases of information.
- **Mail Server** — This gathers e−mail intended for clients on the network and makes it accessible to those clients.
- **FTP Server** — This is used to make files available to users who log in to the server over the network. (Anonymous FTP is a way of making files available to strangers.)
- **Web Server** —This makes Web pages (HTML) and related content available to users on the network.
- **News Server** — This gathers messages from Usenet newsgroups, allowing users to read and respond to topics of interest.

Red Hat Linux can act as any of the server types described above. However, with a growing number of productivity applications becoming available for Linux and friendly X−based desktops (such as KDE and GNOME), Red Hat Linux is beginning to make a better case for becoming a client system as well.

Because Red Hat Linux can run applications as well as offer services, a purer example of a network server is NetWare (from Novell, Inc.). A NetWare server has no way of running applications. It is tuned to store and secure files efficiently, as well as manage groups of printers. With the addition of ZENworks and NDS, NetWare can manage databases of information on all the components on a network.

Peer−to−Peer model

In a peer−to−peer network, computers generally behave as equals. Each computer has most of what it needs to operate on its own. Any computer on the network may offer services to other computers. One computer may share the contents of a CD−ROM, whereas another may offer its printer for use by others.

A typical peer−to−peer network is one where several employees in a business each have a computer at their desk. A shared printer may be connected to one person’s computer, whereas a tape backup system may be connected to another. The drawback with peer−to−peer networks is that if there is a big demand for your printer or other device, your computer’s performance may suffer (and so may your work). That’s why larger networks tend to offload shared services to a server.

After you have made a choice about the typology and computing model you are going to use, the following sections tell you how to actually configure your computers so that they can communicate together.

Setting Up an Ethernet LAN

After you physically install your Ethernet card, it is possible that Red Hat Linux will automatically detect your card and assign it to the first Ethernet interface on your system (eth0). If Red Hat Linux does find your card, after you connect your hardware you don't need to do much more than assign addresses.

Follow this procedure to set up your LAN and install your LAN cards:

1. Choose a network topology. Earlier in this chapter, I described several different network topologies. This procedure assumes that you are using either a star or a bus Ethernet topology.
2. Choose your LAN hardware. You need an Ethernet NIC for each computer on your network, as well as cables that reach from each computer to the hub. Also, you need to purchase a hub. (See the description of these components earlier in this chapter.) Before you purchase a new NIC, read the following discussion on choosing an Ethernet card. Red Hat Linux does not support all NICs. Choosing one that has already been tested with Red Hat Linux can save you some headaches.
3. Install your NIC cards. Power down your computer and physically install the NIC card (following the manufacturer's instructions).
4. Power up your system.

If Red Hat Linux is not installed yet, install the software and reboot (as instructed.) See the discussion on adding Ethernet during Red Hat installation for information on how to answer Ethernet−related questions and the section on configuring host computers for information on adding TCP/IP host names and IP addresses.

When the system comes up, your Ethernet card and interface (eth0) should be ready to use. See the section "Checking Your Ethernet Connection" later in this chapter to learn how to check if your Ethernet connection is working.

Choosing an Ethernet card

Support for many Ethernet cards is available in Red Hat Linux. If you are adding Red Hat Linux to a computer that already has an Ethernet card installed, you can check the list of supported cards or you can just go ahead and install Red Hat. Red Hat Linux may detect the board automatically. If you have a laptop computer that uses a PCMCIA Ethernet card, the card will probably be detected when you boot your computer.

To find out what Ethernet cards are supported, refer to the descriptions of supported networking hardware in the /usr/src/linux*/Documentation/networking directory. To see these descriptions, you need to have the kernel−source package installed.

Recommended Ethernet cards

The Ethernet−HOWTO recommends the following Ethernet cards as being mature and well−tested in Red Hat Linux. If you want to use one of the cards, you just have to make sure that you have the type of card slot in your computer that the card requires. Here are cards for 16−bit ISA and PCI slots:

Recommended 16−Bit ISA Cards:

- SMC−Ultra/EtherEZ
- SMC−Elite (WD80x3)
- 3c509
- Lance
- NE2000

Recommended PCI Cards:

- 3Com Vortex/Boomerang (3c59x/3c9xx)
- DEC Tulip (21xxx)
- Intel EtherExpressPro 100

In addition to these cards, there are many, many cards that work well. You need to gauge the demands on your network to decide if you can get by with a 10 Mbps network or if you need 100 Mbps. Of course, if you are used to a 28.8 Kbps modem connection to the Internet, any working LAN will look fast to you. Even an old 8−bit Ethernet card can provide about 20 times the speed of that old modem. Caution Eight−bit Ethernet cards aren't being made any more. You can, however, find them anywhere that used computer parts are sold. For light performance, you can use wd8003, 3c503, and ne1000 cards. Avoid 3c501 cards because they are said to provide poor performance.

Laptop (PCMCIA) Ethernet cards

There are a large number of Ethernet drivers available for laptop computers (typically using PCMCIA cards). **PCMCIA** stands for **Personal Computer Memory Card International Association**. Essentially, it is a standard that

enables small, removable cards to be used to connect devices to a laptop computer. I have a Netgear 10/100 Mbps PCMCIA card (model FA 410Txc) in my laptop. Red Hat Linux detected my card automatically. I simply configured the eth0 interface using the Network Configuration as described in the discussion on adding your host information after installation later in this chapter. (Basically, what you still need to do is assign an IP address to the interface and decide if you want your LAN connection to start automatically at boot time.)

PCMCIA cards that are supported in Red Hat Linux are defined in the file /etc/pcmcia/config. There are more than 100 PCMCIA Ethernet cards listed, a handful of wireless PCMCIA LAN drivers, and a couple of Token Ring PCMCIA cards. The Ethernet−HOWTO is not up to date with all the PCMCIA Ethernet cards that are now supported. Be sure to check this file before buying a card (or giving up on the one that you have). Note Linux probes for PCMCIA cards by launching the /etc/init.d/pcmcia script. Options required for your PCMCIA cards are contained in the /etc/sysconfig/pcmcia file.

Adding Ethernet after Red Hat is installed

If Red Hat Linux is already installed when you want to add your Ethernet card, simply power down the system, install the card, and reboot the computer. If the card is supported, it is likely that the proper driver will be found and assigned to the board using the eth0 interface. At this point, you simply need to do a bit of configuration (mostly to assign an IP address to the interface) using the Network Configuration window (described in the "Adding host names and IP addresses" section later in this chapter).

Adding two Ethernet cards

If your computer is acting as a router between two LANs (or if it simply is connected to two LANs), you may need to do some special setup to get two LAN cards to work on your computer. If both cards are PCI or EISA LAN cards, they will probably be autodetected so no further configuration will be needed to add the cards. However, if at least one card is an ISA cards, you will need to add some information to your /etc/modules.conf file.

To add two ISA cards to your computer, you need to identify the Ethernet interface associated with each card (eth0, eth1, etc.), then identify the I/O base addresses for each card. Then you must add this information to the /etc/modules.conf file.

The following is an example:

```
alias eth0 3c501
alias eth1 3c503
options 3c501 io=0x280
options 3c503 io=0x300
```

In this example, there is a 3Com 3c501 assigned to the eth0 interface and a 3c503 card assigned to eth1. The base addresses are 0x280 for the 3c501 card and 0x300 for the 3c503 card. The modules are loaded after your computer boots. If both cards you are adding are of the same type, you may be able to use a single options line (for example, options wd io=0x280,0x300) or, if the module supports only one card at a time, you may need two options lines (which results in the module being loaded twice).

Configuring Host Computers

Each computer you communicate with (including your computer) must have a unique address on the network. In TCP/IP, each computer needs to be assigned an IP address and (usually) a host name. When a user runs a program to communicate with another computer, the user typically enters the computer name (or IP address) that it wants to communicate with. There are two basic ways to assign a host name and IP address to the network interfaces:

- **Static Addresses** — With static IP addresses, each computer has its IP address entered in manually. This can be done at Red Hat Linux installation time or later using the Network Configuration window. With this method, the computer has the same IP address each time it boots.
- **Dynamic Addresses** — With dynamic addresses, a client computer gets its IP address assigned from a server on the network when the client boots. The most popular protocol for providing dynamic addresses is called Dynamic Host Configuration Protocol. With this method, a client computer wouldn't necessarily have the same IP address each time it boots.

  For this first example, let's assume that you are setting up a LAN with no outside connections. So, in this section I describe how to use static IP addresses. This is where each computer has a hard IP address and maintains

its own list of host names and IP addresses for the computers it communicates with. This will work fine for communicating with a few computers on a LAN.

Understanding IP addresses

An IP address is a four−part number, with each part represented by a number from 0 to 255 (256 numbers total). Part of that IP address represents the network the computer exists on, whereas the remainder identifies the specific host on that network. Here is an example of an IP address:

```
192.168.35.121
```

Originally, IP addresses were grouped together and assigned to an organization that needed IP addresses, based on IP address classes. Later, a more efficient method, referred to as Classless Inter−Domain Routing (CIDR), was created to improve routing and waste fewer IP addresses.

IP Address Classes

Unfortunately, it's not so easy to understand which part of an IP address represents the network and which represents the host without some explanation of how IP addresses are structured. The way IP addresses are assigned is that a network administrator is given a pool of addresses. The administrator can assign specific host addresses within that pool as new computers are added to the organization's local network. There are three basic classes of IP addresses, each representing a different size network:

- **Class A —** Each Class A address has a number between 0 and 127 as its first part. Host numbers within a Class A network are represented by any combination of numbers in the next three parts. A class A network therefore contains millions of host numbers (approximately 256 X 256 X 256, with a few special numbers being invalid). Whole Class A networks were once assigned only to the largest organizations but, I have been told, are no longer assigned. A valid Class A network number is: `24.`
- **Class B** — A Class B IP address has a number between 128 and 191 in its first part. With a Class B network, however, the second part also represents the network. This enables a Class B network to have more than 64,000 host addresses (256 X 256). A whole Class B network is also rarely assigned. A valid Class B network number is: `135.84`
- **Class C** — A Class C IP address begins with a number between 192 and 223 in its first part. With a Class C network, the first three parts of an IP address represent the network, whereas only the last part represents a specific host. This makes it so each Class C network can have 254 numbers (the numbers 0 and 254 can't be assigned to hosts). Here is an example of a Class C network number: `194.168.1`

  When IP addresses were created, nobody expected that, even though this numbering scheme represented millions of potential addresses, there wouldn't be enough to go around. Now, if you get an official pool of addresses assigned to you for the Internet, you will get either a Class C address or part of a Class A or Class B address. The question becomes: How can a network number be divided among several networks? The answer is: by using a netmask.

Understanding netmasks

Let's say that you are assigned the Class B address 135.84, but you are only given the pool of numbers available to the address 135.84.118. How do you tell your network that every address beginning with 135.84.118 represents a host on your network, but that other addresses beginning with 135.84 should be routed to another network? The answer is with the netmask.

The netmask essentially identifies the network number for a network. When you assign the IP address that is associated with your computer's interface to the LAN (eth0), you are asked for a netmask. By default, your computer will fill in a number that masks the part of your IP address that represents the Class of your network. For example, the default netmasks for Class A, B, and C networks are the following:

- Class A netmask: 255.0.0.0
- Class B netmask: 255.255.0.0
- Class C netmask: 255.255.255.0

Now, if your network was assigned the network number 135.84.118, to tell your computer that 135.84.118 is the network number and not 135.84 (as it normally would be for a Class B address), add a netmask of 255.255.255.0. Thus, your network has available host numbers of 1 to 254 (which would go into the fourth part of the number).

Classless Inter−Domain Routing

The class method of allocating IP addresses had several major drawbacks. First, few organizations fell neatly into one class or another. For most organizations, a Class C address (up to 256 IP addresses) was too small, and a Class B address (up to 65,534 IP addresses) was too big. The result was a lot of wasted numbers in a world where IP addresses were running short. Second, IP classes resulted in too many routing table entries. As a result, routers were becoming overloaded with information.

The Classless Inter−Domain Routing addressing scheme set out to deal with these problems. The scheme is similar to IP address classes, but offers much more flexibility in assigning how much of the 32−bit IP address is the network identifer. Instead of the first 8, 16, or 32 bits identifying the network, 13 to 27 bits could identify the network. As a result, groups of assigned IP addresses could contain from 32 to about 524,000 host addresses.

To indicate the network identifier, a CIDR IP address is followed by a slash (/) and then a number from 13 to 27. A smaller number indicates a network containing more hosts. Here is an example of an IP address that uses the CIDR notation:

```
128.8.27.18/16
```

In this example, the first 16 bits (128.8) represent the network number and the remainder (27.18) represent the specific host number. This network number can contain up to 65,536 hosts (the same as a class B address). The following list shows how many hosts can be represented in networks using different numbers to identify the network:

```
/13 524,288 hosts
/14 262,144 hosts
/15 131,072 hosts
/16 65,536 hosts
/17 32,768 hosts
/18 16,382 hosts
/19 8,192 hosts
/20 4,096 hosts
/21 2,048 hosts
/22 1,024 hosts
/23 512 hosts
/24 256 hosts
/25 128 hosts
/26 64 hosts
/27 32 hosts
```

The CIDR addressing scheme also helps reduce the routing overload problem by having a single, high−level route represent many lower level routes. For example, an Internet service provider could be assigned a single /13 IP network and assign the 500,000−plus addresses to its customers. Routers outside the ISP would only need to know how to reach the ISP for those half−million addresses. The ISP would then be responsible for maintaining routing information for all of the host routes with that network address.

Getting IP addresses

The impact of assigning IP addresses for the computers on your LAN. How you choose which IP addresses to use depends on your situation. If you are part of a large organization, you should get addresses from the network administrator of your organization. Even if you don't connect to other LANs in your organization, having unique addresses can make it easier to connect to other LANs in the future. If you are setting up a network for yourself (with no other networks to consider in your organization), use private addresses or (if you need the network to be part of the Internet) apply for your own domain name and IP addresses. If you don't need to have your LAN accessible from the Internet, choose IP addresses from the set of available general−purpose IP addresses. The private IP addresses not used on any public part of the Internet.

Network Class	Network Numbers	Addresses per Network Number
Class A	10.0.0.0	167,777,216
Class B	172.16.0.0 to 172.31.0.0	65,536
Class C	192.168.0.0 to 192.168.255.255	256

Adding host names and IP addresses

When you install Red Hat Linux, you are given the opportunity to add your TCP/IP host name and IP address, as well as some other information, to your computer. You also need to set up a way to reach the computers on your LAN.

That's done either by adding all computer names and IP addresses to your /etc/hosts file (described here) or by using a DNS server.

If you did not identify your IP address during installation of Red Hat Linux, you can do so at a later time using the Network Configuration window (neat command). This procedure lets you attach a particular IP address to your Ethernet interface, so your computer knows what address to listen for. Note A computer can have more than one IP address because it can have more than one network interface. Each network interface must have an IP address (even if that address is assigned only temporarily). So, if you have two Ethernet cards on your computer (eth0 and eth1), each needs its own IP address. Also, the address 127.0.0.1 represents the local host, so users on the local computer can access services in loopback. To define your IP address for your eth0 interface, follow this procedure:

1. Start the Network Configuration. As root user from a Terminal window, type **neat**. The Network Configuration window appears.
2. Click the Devices tab. A listing of your existing network interfaces appears.
3. Double−click the eth0 interface. A pop−up window appears, enabling you to configure your eth0 interface. The Network Configuration window and the pop−up Ethernet Device window configuring eth0.
4. Select "Activate device when computer starts" to have the network interface start at boot time.
5. Click the Protocols tab. You should at least see TCP/IP.
6. Click TCP/IP and select Edit. On the TCP/IP Settings window that appears, you can enter the following information:
    - **Automatically obtain IP address settings with:** Make sure that the check box is off next to this box. (If you were getting your IP address from a DHCP server, this box would be on and the rest of the information would be obtained automatically.)
    - **Address:** Type the IP address of this computer into the Address box. This number must be unique on your network.
    - **Subnet Mask:** Enter the netmask to indicate what part of the IP address represents the network. (Netmask is described earlier in this chapter.)
    - **Default Gateway Address:** If there is a computer or router connected to your LAN that is providing routing functions to the Internet or other network, type the IP address of the computer into this box.
7. Click OK in the TCP/IP Settings window to save the current configuration.
8. Click OK in the Ethernet Device window to close that window.
9. Click Apply in the Network Configuration window to apply the changes.

Checking Your Ethernet Connection

After your LAN has been set up, your Ethernet cards installed, and host names and addresses added, there are several methods you can use to check that everything is up and working. For example, you can check your boot messages to make sure that your board was detected and that you can use the ping command to make sure you can connect to other computers.

Did Linux find your Ethernet driver at boot−time?

Type the following to check that Linux found your card and installed the Ethernet interface properly:

```
dmesg | grep eth
```

The dmesg command lists all the messages that were output by Linux at boot−time. The grep eth command causes only those lines that contain the word *eth* to be printed. The first message shown below appeared on my laptop computer with the Netgear card. The second example is from my computer with the EtherExpress Pro/100 card:

```
eth0: NE2000 Compatible: port 0x300, irq3, hw_addr 00:80:C8:8C:8E:49
eth0: OEM i82557/i82558 10/100 Ethernet at 0xccc0, 00:90:27:4E:67:35, IRQ 17.
```

The message in the first example shows that a card was found at IRQ3 with a port address of 0x300 and an Ethernet hardware address of 00:80:C8:8C:8E:49. In the second example, the card is at IRQ 17, the port address is 0xccc0, and the Ethernet address is 00:90:27:4E:67:35. If the eth0 interface is not found, but you know that you have a supported Ethernet card, check that your Ethernet card is properly seated in its slot.

Can you reach another computer on the LAN?

Try communicating with another computer on the LAN. The ping command can be used to send a packet to another computer and to ask for a packet in return. You could give ping either a host name (pine) or an IP address (10.0.0.10). For example, to ping a computer on the network called pine, type the following command:

```
# ping pine
```

If the computer can be reached, the output will look similar to the following:

```
PING pine.trees (10.0.0.10): 56 data bytes
64 bytes from 10.0.0.10: icmp_seq=0 ttl=255 time=0.6 ms
64 bytes from 10.0.0.10: icmp_seq=1 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=7 ttl=255 time=0.6 ms
64 bytes from 10.0.0.10: icmp_seq=8 ttl=255 time=0.5 ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=255 time=0.5 ms
--- pine.trees ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.6 ms
```

A line of output is printed each time a packet is sent and received in return. It shows how much data was sent and how long it took for each package to be received. After you have watched this for a while, type Ctrl+C to stop ping. At that point, it will show you statistics of how many packets were transmitted, received, and lost. If you don't see output that shows packets have been received, it means that you are not contacting the other computer. Try to verify that the names and addresses of the computers that you want to reach are in your /etc/hosts file or that your DNS server is accessible. Next, confirm that the names and IP addresses you have for the other computers that you are trying to reach are correct (the IP addresses are the most critical).

Is your Ethernet connection up?

Using the **ifconfig** command, you can determine whether your Ethernet (and other network interfaces) are up and running. Type the following command:

```
# ifconfig
```

The output that appears will be similar to the following:

```
eth Link encap:Ethernet HWaddr 00:90:27:4E:67:35
inet addr:10.0.0.11 Bcast:10.255.255.255 Mask:255.0.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:156 errors:0 dropped:0 overruns:0 frame:0
TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:20179 (19.7 Kb) TX bytes:19960 (19.4 Kb)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:3924 Metric:1
RX packets:56 errors:0 dropped:0 overruns:0 frame:0
TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:3148 (3.0 Kb) TX bytes:3148 (3.0Kb)
```

In this example, there are currently two network interfaces up on the current computer. The first section shows your Ethernet interface (eth0) and its hardware address, Ethernet hardware address, IP address (inet addr), broadcast address, and network mask. The next lines provide information on packets that have been sent, along with the number of errors and collisions that have occurred. Note The lo entry is for loopback. This enables you to run TCP/IP commands on your local system without having a physical network up and running.

If your eth0 interface does not appear, it may still be configured properly, but not running at the moment. Try to start the eth0 interface by typing the following:

```
# ifconfig eth0 up        (enable)
```

After this, type ifconfig again to see if eth0 is now running. If it is, it may be that eth0 is simply not configured to start automatically at boot time. You can change it so Ethernet starts at boot−time (which I recommend), using the Network Configuration window (described earlier in this chapter).

Tip If your network interfaces are not running at all, you can try to start them from the network initialization script. This interface reads parameters and basically runs ifconfig for all network interfaces on your computer. Type the following to restart your network:

```
# /etc/init.d/network restart
```

Another way to see statistics for your Ethernet driver is to list the contents of the process pseudo file system for network devices. To do that, type the following:

```
# cat /proc/net/dev
```

The output should look similar to the following:

```
Inter-| Receive | Transmit
face |bytes packets errs drop fifo frame compressed multicast|bytes
packets errs drop fifo colls carrier compressed
lo: 5362 64 0 0 0 0 0 0 5362 64 0 0 0 0 0 0
eth0: 3083 35 0 0 0 0 0 0 3876 31 0 0 0 0 0 0
```

The output is a bit hard to read (our book isn't wide enough to show it without wrapping around). With this output, you can see Receive and Transmit statistics for each interface. This output also shows you how many Receive and Transmit errors occurred during communication.

Using Dial−up Connections to the Internet

Most individuals and even many small businesses that need to connect to the Internet do so using modems and telephone lines. Your modem connects to a serial port (COM1, COM2, and so on) on your computer and then into a telephone wall jack. Then your computer dials a modem at your Internet Service Provider or business that has a connection to the Internet.

The two most common protocols for making dial−up connections to the Internet (or other TCP/IP network) are Point−to−Point Protocol (PPP) and Serial Line Internet Protocol (SLIP). Of the two, PPP is more popular and more reliable. SLIP, however, has been around longer.

Getting information

To establish a PPP connection, you need to get some information from the administrator of the network that you are connecting to. This is either your Internet Service Provider (ISP) when you sign up for Internet service or the person who walks around carrying cables, a cellular phone, and a beeper where you work (when a network goes down, these people are in demand!). Here is the kind of information you need to set up your PPP connection:

- **PPP or SLIP** — Does the ISP require SLIP or PPP protocols to connect to it? In this book, I describe how to configure PPP.
- **Telephone number** — This telephone number gives you access to the modem (or pool of modems) at the ISP. If it is a national ISP, make sure that you get a local telephone number (otherwise, you will rack up long distance fees on top of your ISP fees).
- **Account name and password** — This information is used to verify that you have an Internet account with the ISP. This is typically used when you connect to Red Hat Linux or other UNIX system. (When connecting to an NT server, the account name may be referred to as a system name.)
- **An IP number** — Most ISPs use Dynamic IP numbers, which means that you are assigned an IP number temporarily when you are connected. Your ISP assigns a permanent IP number if it uses Static IP addresses. If your computer or all the computers on your LAN need to have a more permanent presence on the network, you may be given one Static IP number or a set of Static IP numbers to use.
- **DNS IP numbers** — When you use a Web browser, FTP utility, or other Internet program to request a service from a computer on the network, you need a way to translate that name (for example, whatever.com) into an Internet address. Your computer will do this by querying a Domain Name System (DNS) server. Your ISP should give you at least one, and possibly two or three, IP addresses for a primary (and possibly secondary and tertiary) DNS server.
- **Mail server** — If your ISP is providing you with an e−mail account, you need to know the address of the mail server, the type of mail service (such as Post Office Protocol or POP), and the authentication password for the mail server in order to get your e−mail.
- **News server** — To be able to participate in newsgroups, the ISP may provide you with the hostname of a news server. If the server requires you to log on, you will also need a password. After you have gathered this information, you are ready to set up your connection to the Internet. To configure Red Hat Linux to connect to your ISP, follow the PPP procedure described below.

Setting up dial−up PPP

Point−to−Point Protocol (PPP) is used to create Internet Protocol (IP) connections over serial lines. Most often, the serial connection is established over a modem; however, it will also work over serial cables (null modem cables) or digital lines (including ISDN and DSL digital media). PPP is a common way to connect an individual computer or LAN to a TCP/IP Wide Area Network (such as the Internet). Although one side must dial out while the other side must receive the call to create a PPP connection over a modem, after the connection is established, information can flow in both directions. For the sake of clarity, however, I refer to the computer placing the call as the client and the computer receiving the call as the server. To simplify the process of configuring PPP (and other network interfaces), Red Hat Linux provides a dial−up configuration tool for both the GNOME and KDE interfaces. Those interfaces are, respectively, as follows:

- **Dial-up Configuration Tool** — From the GNOME desktop menu, choose Programs → Internet → Dialup Configuration. The Internet Connection window that appears lets you configure and test your dial−up PPP connection.
- **Kppp Window** — From the KDE desktop menu, choose Internet → Internet Dialer. This runs the kppp command. From the kppp window you can set up a PPP dial−up connection and launch it. Before you begin either of the two dial−up procedures, physically connect your modem to your computer, plug it in, and connect it to your telephone line. If you have an internal modem, you will probably see a telephone port on the back of your computer that you need to connect. After the modem is connected, reboot Red Hat Linux so it can automatically detect and configure your modem.

Creating a dial−up connection from GNOME

To configure dial−up networking from the GNOME desktop, you should use the Dialup Configuration window. To start it, choose Programs → Internet → Dialup Configuration from the GNOME menu. A connection wizard appears to help you configure your PPP dial−up connection Follow the procedure below from the first Dialup Configuration Tool window to configure your dial−up connection.

1. From the Add New Internet Connection window that appears, click Next to continue. If you do not have a modem configured, you are asked if you want to configure one.
2. Click Next to configure a modem. A pop−up window searches your computer for a modem. If it finds your modem, its location will be filled in on the Enter a modem window. Otherwise, you will have to enter the location of the modem yourself.
3. Select the modem you want from the list of modems found (there will probably only be one). Click "Keep this modem," and then click Next to continue. A window appears, asking for your account name and telephone number.
4. Enter the account name (any name to identify the account) and the telephone number of the ISP you want to dial into. Then click Next to continue. (The optional Prefix is in case you need to dial a 9 or some other number to get an outside dial tone before dialing.) The window asks for your user account name and password.
5. Type in the account name and password. You should have received this information from your ISP. The ISP may have called the account name a Login ID or similar name. Click Next to continue. The Other Options window appears.
6. Select Normal ISP (unless your ISP happens to appear in the listing, in which case select it instead). Then click Next to continue. The Create the account window appears.
7. If all the information looks correct, click Finish (otherwise, click the Back button to change any information). The completed connection type appears in the Internet Connections window.
8. To test your connection, select your new PPP account and click the Debug button. The Internet dialer starts up and dials your ISP.

If everything is working properly, you should see your login and password accepted and the PPP connection completed. Try opening Netscape Communicator and see if you can access a Web site on the Internet. If this doesn't work the first time, don't be discouraged. There are many things to check to get your dial−up PPP connection working. Skip ahead to the "Checking your PPP connection" section.

Creating a dial−up connection from KDE

To configure a dial−up PPP connection from the KDE desktop, you can use the kppp window. To open that window, choose Internet → Internet Dialer from the KDE menu. Then click the Setup button.

Note Instead of using the Internet Dialer (kppp) window, you can use the Dialup Configuration window described in the section "Creating a dial−up connection from GNOME." Open the Dialup Configuration window from the KDE desktop by selecting Internet → Dialup Configuration from the KDE menu.

1. From the kppp Configuration window (Accounts tab), click New. A pop−up window asks if you want to use the wizard to create a new account.
2. Click Dialog Setup. A New Account window appears.
3. From the Dial tab on the New Account window, add the following information:
   - **Connection Name** — Enter any name you choose to identify the connection. Typically, the name would identify your ISP.
   - **Phone Number** — Click on the Add button, enter the telephone number of the ISP's modem pool, and click on OK.
   - **Authentication** — Determine from your ISP the type of authentication that is used to establish the connection. Many ISPs use a PAP or CHAP type of authentication (which are used with Windows NT and other types of servers), while universities and other sites where UNIX and Linux servers are used tend to use Terminal and Script−based authentication.
   - **Customize pppd arguments** — Click on this button, type an argument you want to add, click on Add, repeat for additional arguments (optional), and click on OK. These arguments are passed to the pppd daemon (which establishes and maintains your PPP connections). Some of these arguments are described later in the section "Checking your PPP connection."
4. Click the IP tab. Chances are that the ISP will use Dynamic IP addresses. If the ISP gave you a Static IP address, click the Static IP Address box and type in the address and netmask the ISP gave you. You can also click on the "Autoconfigure hostname..." box to have your host name automatically assigned from your ISP.
5. Click the DNS tab. This is where you enter your domain name and the IP address for the DNS server (which is used to resolve Internet host/domain names into IP addresses). If DNS servers are not assigned dynamically (which they probably are), you will typically be given two DNS servers to enter (a primary and a backup).
6. Click the Login Script tab. This is a somewhat advanced feature. It can be used if your dial−up ISP connection doesn't do the standard PAP, CHAP, or terminal login ways of setting up a connection. If that is the case, you can set up a custom "chat" script here that defines what you expect to receive from the remote side and what you will send in response. (When you try your connection a few steps later, you will be able to watch this chat take place.)
7. Click the Execute tab. If you want to run a special command or script before or at the point of connection or disconnection, you can add the full path to the command or script in the appropriate box. (You will typically leave these blank.)
8. Click the Accounting tab. If you need to account for the amount of traffic being received or sent over this connection, you can click the Enable Accounting button on this tab. You must then select the Available rules, based on your country and type of service. This feature is more useful outside of the United States, where billing for Internet service is done differently.
9. Click OK. The new account should appear in the Account Setup box.
10. Click the Device tab. Select the modem device that will be used for the connection. Your modem may already be linked to the /dev/modem device (whether it is on COM1 or COM2). To specifically set the modem to one of those ports, you could select /dev/ttyS0 for COM1, or select /dev/ttyS1 for COM2 (and so on).
11. Click OK to exit from the kppp Configuration window.
12. From the main kppp window (which should still be on your screen), make sure that your new connection type appears in the Connect to window. The first time you try the connection, click the Show Log Window box. Type the login ID and password for your ISP account.
13. Click Connect. The Login Script Debug window will step through the process of initializing the modem, dialing, and making the PPP connection. If all goes well, you should be able to start browsing the Internet. If the connection fails, skip to the "Checking your PPP connection" section for information on hunting down the problem.

Launching your PPP connection

After you have a working PPP connection configured, you can set up that connection to launch easily from the desktop. Here's how: From the GNOME desktop:

1. Right−click Panel and then choose Panel → Add to Panel → Applet → Network → RH PPP Dialer from the GNOME menu. When the Choose pop−up window appears, click the interface you want to use and then click OK.
2. You can either start the connection now or not. In either case, after you finish Step 1, an icon appears on the panel that you can click to immediately connect to the ISP (click the green button). From the KDE desktop:
    a. Right−click the desktop and choose Create New → Link to Application.
    b. Type **Dialup.kdelnk**.
    c. Click the icon and select an icon to represent the application; then click OK (There is one called kppp that you can use.)
    d. Click the Execute tab and then type **kppp** into the Execute box. Click OK.
    e. An icon called Dialup appears on your KDE desktop. Click it to open the kppp window. Then select your ISP from the Connect box and click Connect to start your PPP connection. From this point forward, icons will appear on your desktop that you can select to immediately connect to your ISP over the dial−up connection you configured. Both GNOME and KDE support drag−and−drop, so you can drag the dial−up icon to the desktop to make it even more easily available.

The netstat Command

netstat is a useful tool for checking your network configuration and activity. It is in fact a collection of several tools lumped together. We discuss each of its functions in the following sections.

Displaying the Routing Table

When you invoke netstat with the -r flag, it displays the kernel routing table in the way we've been doing with route. On vstout, it produces:

```
# netstat -r
Kernel IP routing table
Destination Gateway     Genmask           Flags       MSS Window irtt Iface
127.0.0.1   *           255.255.255.255   UH          0     0     0     lo
172.16.1.0  *           255.255.255.0     U           0     0     0     eth0
172.16.2.0  172.16.1.1  255.255.255.0     UG          0     0     0     eth0
```

The -n option makes netstat print addresses as dotted quad IP numbers rather than the symbolic host and network names. This option is especially useful when you want to avoid address lookups over the network .

The second column of netstat's output shows the gateway to which the routing entry points. If no gateway is used, an asterisk is printed instead. The third column shows the "generality" of the route, i.e., the network mask for this route. When given an IP address to find a suitable route for, the kernel steps through each of the routing table entries, taking the bitwise AND of the address and the genmask before comparing it to the target of the route.

The fourth column displays the following flags that describe the route:

G The route uses a gateway.
U The interface to be used is up.
H Only a single host can be reached through the route. For example, this is the case for the loopback entry 127.0.0.1.
D This route is dynamically created. It is set if the table entry has been generated by a routing daemon like gated or by an ICMP redirect message.
M This route is set if the table entry was modified by an ICMP redirect message.
! The route is a reject route and datagrams will be dropped.

The next three columns show the MSS, Window and irtt that will be applied to TCP connections established via this route. The MSS is the Maximum Segment Size and is the size of the largest datagram the kernel will construct for

transmission via this route. The Window is the maximum amount of data the system will accept in a single burst from a remote host. The acronym irtt stands for "initial round trip time." The TCP protocol ensures that data is reliably delivered between hosts by retransmitting a datagram if it has been lost. The TCP protocol keeps a running count of how long it takes for a datagram to be delivered to the remote end, and an acknowledgement to be received so that it knows how long to wait before assuming a datagram needs to retransmitted;

this process is called the round-trip time. The initial round-trip time is the value that the TCP protocol will use when a connection is first established. For most network types, the default value is okay, but for some slow networks, notably certain types of amateur packet radio networks, the time is too short and causes unnecessary retransmission. The irtt value can be set using the route command. Values of zero in these fields mean that the default is being used.

Displaying Interface Statistics

When invoked with the -i flag, netstat displays statistics for the network interfaces currently configured. If the -a option is also given, it prints all interfaces present in the kernel, not only those that have been configured currently. On vstout, the output from netstat will look like this:

```
# netstat -i
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flags
lo    0     0 3185      0     0     0     3185  0     0           0 BLRU
eth0  1500  0 972633      17     20     120    628711 217    0           0 BRU
```

The MTU and Met fields show the current MTU and metric values for that interface. The RX and TX columns show how many packets have been received or transmitted error-free (RX-OK/TX-OK) or damaged (RXERR/ TX-ERR); how many were dropped (RX-DRP/TX-DRP); and how many were lost because of an overrun (RX-OVR/TX-OVR).

The last column shows the flags that have been set for this interface. These characters are one-character versions of the long flag names that are printed when you display the interface configuration with ifconfig:

B	A broadcast address has been set.
L	This interface is a loopback device.
M	All packets are received (promiscuous mode).
O	ARP is turned off for this interface.
P	This is a point-to-point connection.
R	Interface is running.
U	Interface is up.

Displaying Connections

netstat supports a set of options to display active or passive sockets. The options -t, -u, -w, and -x show active TCP, UDP, RAW, or Unix socket connections. If you provide the -a flag in addition, sockets that are waiting for a connection (i.e., listening) are displayed as well. This display will give you a list of all servers that are currently running on your system. Invoking netstat -ta on vlager produces this output:

```
$ netstat -ta
Active Internet Connections
Proto Recv-Q Send-Q     Local Address Foreign Address       (State)
tcp   0           0     *:domain          *:*               LISTEN
tcp   0           0     *:time            *:*               LISTEN
tcp   0           0     *:smtp            *:*               LISTEN
tcp   0           0     vlager:smtp       vstout:1040       ESTABLISHED
tcp   0           0     *:telnet          *:*               LISTEN
tcp   0           0     localhost:1046    vbardolino:telnet ESTABLISHED
tcp   0           0     *:chargen         *:*               LISTEN
tcp   0           0     *:daytime         *:*               LISTEN
tcp   0           0     *:discard         *:*               LISTEN
tcp   0           0     *:echo            *:*               LISTEN
tcp   0           0     *:shell           *:*               LISTEN
tcp   0           0     *:login           *:*               LISTEN
```

This output shows most servers simply waiting for an incoming connection. However, the fourth line shows an incoming SMTP connection from vstout, and the sixth line tells you there is an outgoing telnet connection to vbardolino.39

The netconfig command

Contains the netconfig utility, which is a graphical tool for configuring network interfaces.

SLIP Operation

Dialup IP servers frequently offer SLIP service through special user accounts. After logging in to such an account, you are not dropped into the common shell; instead, a program or shell script is executed that enables the server's SLIP driver for the serial line and configures the appropriate network interface. Then you have to do the same at your end of the link.

On some operating systems, the SLIP driver is a user-space program; under Linux, it is part of the kernel, which makes it a lot faster. This speed requires, however, that the serial line be converted to the SLIP mode explicitly. This conversion is done by means of a special tty line discipline, SLIPDISC. While the tty is in normal line discipline (DISC0), it exchanges data only with user processes, using the normal read(2) and write(2) calls, and the SLIP driver is unable to write to or read from the tty. In SLIPDISC, the roles are reversed: now any user-space processes are blocked from writing to or reading from the tty, while all data coming in on the serial port is passed directly to the SLIP driver.

The SLIP driver itself understands a number of variations on the SLIP protocol. Apart from ordinary SLIP, it also understands CSLIP, which performs the so-called Van Jacobson header compression (described in RFC-1144) on outgoing IP packets. This compression improves throughput for interactive sessions noticeably. There are also six-bit versions for each of these protocols.

A simple way to convert a serial line to SLIP mode is by using the slattach tool. Assume you have your modem on /dev/ttyS3 and have logged in to the SLIP server successfully. You will then execute:

```
# slattach /dev/ttyS3 &
```

This tool switches the line discipline of ttyS3 to SLIPDISC and attaches it to one of the SLIP network interfaces. If this is your first active SLIP link, the line will be attached to sl0; the second will be attached to sl1, and so on. The current kernels support a default maximum of 256 simultaneous SLIP links. The default line discipline chosen by slattach is CSLIP. You may choose any other discipline using the –p switch. To use normal SLIP (no compression), you use:

```
# slattach -p slip /dev/ttyS3 &
```

A special pseudo-discipline is available called adaptive, which causes the kernel to automatically detect which type of SLIP encapsulation is being used by the remote end.

UUCP Networks

Unix-to-Unix Copy (UUCP) started out as a package of programs that transferred files over serial lines, scheduled those transfers, and initiated execution of programs on remote sites. It has undergone major changes since its first implementation in the late seventies, but it is still rather spartan in the services it offers. Its main application is still in Wide Area Networks, based on periodic dialup telephone links.

UUCP was first developed by Bell Laboratories in 1977 for communication between their Unix development sites. In mid-1978, this network already connected over 80 sites. It was running email as an application, as well as remote printing. However, the system's central use was in distributing new software and bug fixes. Today, UUCP is not confined solely to the Unix environment. There are free and commercial ports available for a variety of platforms, including AmigaOS, DOS, and Atari's TOS.

One of the main disadvantages of UUCP networks is that they operate in batches. Rather than having a permanent connection established between hosts, it uses temporary connections. A UUCP host machine might dial in to another UUCP host only once a day, and then only for a short period of time. While it is connected, it will transfer all of the news, email, and files that have been queued, and then disconnect. It is this queuing that limits the sorts of applications that UUCP can be applied to. In the case of email, a user may prepare an email message and post it. The message will stay queued on the UUCP host machine until it dials in to another UUCP host to transfer the message. This is fine for network services such as email, but is no use at all for services such as

rlogin.

Despite these limitations, there are still many UUCP networks operating all over the world, run mainly by hobbyists, which offer private users network access at reasonable prices. The main reason for the longtime popularity of UUCP was that it was very cheap compared to having your computer directly connected to the Internet. To make your computer a UUCP node, all you needed was a modem, a working UUCP implementation, and another UUCP node that was willing to feed you mail and news. Many people were prepared to provide UUCP feeds to individuals because such connections didn't place much demand on their existing network. We cover the configuration of UUCP in a chapter of its own later in the book, but we won't focus on it too heavily, as it's being replaced rapidly with TCP/IP, now that cheap Internet access has become commonly available in most parts of the world.

Configuring a Mail Server

Simple Mail Transport Protocol (SMTP) is the TCP/IP mail transport protocol. Linux provides full SMTP support through the sendmail program, although sendmail does more than just send and receive SMTP mail. sendmail provides mail aliases and acts as a "mail router," routing mail from all of the different user mail programs to the various mail delivery programs while ensuring that the mail is properly formatted for delivery.

Sendmail configurations are built using the m4 macro processing language. The output of the m4 process is the sendmail.cf file, which is the configuration file read by sendmail. To fully understand and manage sendmail, you need to understand its functions, the sendmail.cf file from which it reads its configuration, and the m4 macros used to build that file.

Using Mail Aliases

Mail aliases are defined in the aliases file. The location of the aliases file is set in the sendmail configuration file. On Linux systems, the file is usually located in the /etc directory (/etc/aliases), and it is occasionally located in the /etc/mail directory. The basic format of entries in the file is

alias: recipient

The alias is the username in the e−mail address, and recipient is the name to which the mail should be delivered. The recipient field can contain a username, another alias, or a final delivery address. Additionally, there can be multiple recipients for a single alias. sendmail aliases perform important functions that are an essential part of creating a mail server. Mail aliases do the following:

Specify nicknames for individual users Nicknames can be used to direct mail addressed to special names, such as postmaster or root, to the real users that do those jobs.

Forward mail to other hosts sendmail aliases automatically forward mail to the host address included as part of the recipient address.

Define mailing lists An alias with multiple recipients is a mailing list.

A Sample *aliases* File

```
#
# @(#)aliases 8.2 (Berkeley) 3/5/94
#
# Aliases in this file will NOT be expanded in the header from
# Mail, but WILL be visible over networks or from /bin/mail.
#
# >>>>>>>>>> The program "newaliases" must be run after
# >> NOTE >> this file is updated for any changes to
# >>>>>>>>>> show through to sendmail.
#
# Basic system aliases -- these MUST be present.
mailer-daemon: postmaster
postmaster: root
# General redirections for pseudo accounts.
bin: root
daemon: root
adm: root
lp: root
sync: root
shutdown: root
halt: root
mail: root
```

```
news: root
uucp: root
operator: root
games: root
gopher: root
ftp: root
nobody: root
apache: root
named: root
xfs: root
gdm: root
mailnull: root
postgres: root
squid: root
rpcuser: root
rpc: root
ingres: root
system: root
toor: root
manager: root
dumper: root
abuse: root
newsadm: news
newsadmin: news
usenet: news
ftpadm: ftp
ftpadmin: ftp
ftp-adm: ftp
ftp-admin: ftp
webmaster: root
# trap decode to catch security attacks
decode: root
# Person who should get root's mail
root: staff
# System administrator mailing list
staff: kathy, craig, david@parrot, sara@hawk, becky@parrot
owner-staff: staff-request
staff-request: craig
# User aliases
norman.edwards: norm
edwardsn: norm
norm: norm@hawk.foobirds.org
rebecca.hunt: becky@parrot
andy.wright: andy@falcon.foobirds.org
sara.henson: sara@hawk
kathy.McCafferty: kathy
kathleen.McCafferty: kathy
```

The Red Hat /etc/aliases file opens with several comment lines. Ignore the information about which mail programs display aliases in the headers of mail messages; it is not really significant. The comment that is significant is the one that tells you to run newaliases every time you update this file. sendmail does not read the /etc/aliases file directly. Instead, it reads a database file produced from this file by the newaliases command.

The *sendmail* Configuration File

The file that defines the sendmail runtime configuration is sendmail.cf, which is a large, complex file that is divided into seven different sections. The file is so large and so complex that system administrators are often intimidated by it. You needn't be. The file is designed to be easily parsed by sendmail, not to be easily written by a system administrator. But normally, you don't directly write to this file. Instead, you build the file with the m4 commands. It is important to have a basic understanding of the syntax and structure of the sendmail.cf file in order to better understand the effect of the m4 commands and to gain the mastery needed for troubleshooting.

The section labels from the Red Hat sendmail.cf file provide an overview of the structure and the function of the file.

- **Local Info** This section defines the configuration information specific to the local host.
- **Options** This section sets the options that define the sendmail environment.
- **Message Precedence** This section defines the sendmail message precedence values.
- **Trusted Users** This section defines the users who are allowed to change the sender address when they are sending mail.

- **Format of Headers** This section defines the headers that sendmail inserts into mail.
- **Rewriting Rules** This section holds the commands that rewrite e−mail addresses from user mail programs into the form required by the mail−delivery programs.
- **Mailer Definitions** This section defines the programs used to deliver the mail. The rewrite rules used by the mailers are also defined in this section.

Configuring the *sendmail.cf* File

It's important to realize how rarely the sendmail.cf file needs to be modified on a typical Linux system. The configuration file that comes with your Linux system will work. Generally, you modify the sendmail configuration not because you need to, but because you want to. You modify it to improve the way things operate, not to get them to operate. To illustrate this, let's look at the default Red Hat configuration on the system parrot.foobirds.org.

Using the default configuration, the From address on outbound e−mail is user@parrot.foobirds.org. This is a valid address, but assume that it's not exactly what you want. In the last chapter, you defined MX records for the domain. To use them, you want people to use addresses in the form *user*@foobirds.org, so you don't want the hostname in outbound e−mail addresses. To create the new configuration, you need to understand the purpose of class M and macro M, both of which are found in the Local Info section of the sendmail.cf file.

sendmail calls hiding the real hostname *masquerading*. Thus, the name of the macro used to rewrite the sender host address is M. Set M to the domain name to replace the name of the local host in outbound mail with the name of the domain. Class M defines other hostnames, not just the local hostname, that also should be rewritten to the value of macro M. Class M is used on mail servers that need to rewrite sender addresses for their clients.

Checking the Red Hat sendmail.cf file on parrot, you find that no value is assigned to macro M, which means that masquerading is not being used. Further, you find that there is no class M declaration in the file. To masquerade the local host as foobirds.org and to masquerade the outbound mail from the clients robin and puffin, copy the sendmail.cf file to test.cf and then edit test.cf, changing the macro M declaration and adding a class M declaration:

```
# who I masquerade as (null for no masquerading)
DMfoobirds.org
# class M: host names that should be converted to $M
CMpuffin.foobirds.org robin.foobirds.org
```

Given these macro M and class M definitions, parrot rewrites its own outbound mail to *user*@foobirds.org, as well as rewriting mail from *user*@puffin.foobirds.org or user@robin.foobirds.org to user@foobirds.org. parrot is a mail server. Although you might use macro M on any system, you won't use class M on any type of system except a mail server.

A problem with using class M is that kathy@puffin.foobirds.org, kathy@robin.foobirds.org, and kathy@parrot.foobirds.org are all rewritten as kathy@foobirds.org. That's great if there really is only one kathy in the entire domain; otherwise, this may not be what you want. Coordinate usernames carefully across all systems. It simplifies the configuration of several different applications.

After setting a value for the M macro in the test.cf file, run a test to see if it works. Running sendmail with the test configuration does not affect the sendmail daemon that was started by the boot script. A separate instantiation of sendmail is used for the test.

Testing Your New Configuration

Test whether or not the change made to macro M in the configuration files modifies the rewrite process by directly testing the rewrite rulesets. First, you need to find out what rules are used to rewrite the address.

It is followed by different rulesets, depending on whether the address is a delivery address, a sender address, or a recipient address. Furthermore, the rulesets used for sender and recipient addresses vary, depending on the mailer that is used to deliver the mail. All addresses are then processed by ruleset final.

There are two variables that determine the rulesets used to process an address: the type of address and the mailer through which it is processed. The three address types are delivery address, recipient address, and sender address. You know the address type because you select the address being tested. In the example, the concern is the sender address.

There are two types of sender addresses: the sender address in the message header and the sender address in the "envelope." The message header address is the one on the From line sent with the mail. You probably see it in the mail headers when you view the message with your mail reader. The "envelope" address is the address used during the SMTP protocol interactions. The one that we're interested in is the one that remote users see in the mail—the header address.

Using *m4* to Configure *sendmail*

The sendmail distribution contains m4 source files that build the sendmail.cf file. Sample m4 source files probably are included with your Linux system. If your Linux distribution doesn't include the m4 source files, where they are stored as part of the latest sendmail distribution.

This section builds a custom sendmail.cf file using the m4 source files that come with a Red Hat system. On a Red Hat system, the m4 source files are in an RPM package separate from the package that includes the sendmail program.

The Linux OSTYPE File

The OSTYPE file contains operating system–specific configuration values. The most common configuration variation between the different operating systems that run sendmail is the location of files. Variables that define pathnames are commonly stored in the OSTYPE file. However, any valid m4 macro can be placed in the OSTYPE file.

The command OSTYPE(linux) loads a file named linux.m4 from the ostype directory. On our sample Red Hat system, the full path of this directory is /usr/share/sendmail-cf/ostype.

The *linux.m4* OSTYPE File

```
----------------------------------
divert(0)
VERSIONID(`$Id: linux.m4,v 8.11.16.2 2000/09/17 17:04:22 gshapiro Exp $')
define(`confEBINDIR', `/usr/sbin')
ifdef(`PROCMAIL_MAILER_PATH',,
define(`PROCMAIL_MAILER_PATH', `/usr/bin/procmail'))
FEATURE(local_procmail)
```

The Apache Web Server

Linux systems make excellent web servers. In fact, the Apache server software that comes with Linux is the most widely used web server in the world. The daemon that Apache installs on a Linux system to create a web server is the Hypertext Transport Protocol daemon (httpd).

Running *httpd*

After the Apache RPM is installed, use a tool such as chkconfig or tksysv to add httpd to the boot process to ensure that the server restarts when the system reboots. For example, to start httpd for runlevels 3 and 5 on a Red Hat system, enter the following chkconfig command:

```
[root]# chkconfig --level 35 httpd on
[root]# chkconfig --list httpd
httpd 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

If your system doesn't have chkconfig, use another tool, such as tksysv. tksysv is used to run httpd at startup. Highlight httpd in the Available box, click Add, and then click Done in the next two dialog boxes to add it to the startup process.

Configuring the Apache Server

You edit the httpd.conf file to set the web administrator's e-mail address in ServerAdmin and the server's hostname in ServerName. Beyond that, the httpd configuration provided with a Linux distribution should be adequate for that version of Linux. The httpd.conf file is stored in the /etc/httpd/conf directory on Red Hat systems. Another operating system may place the configuration file in a different directory. To find where it

is located on your system, look in the script that was used to start httpd. The location of the httpd.conf file is defined there. The locations of other files used by httpd are defined in httpd.conf. Another very simple way to locate the file is with the find command:

```
# find / -name httpd.conf -print
```

This command tells find to search every directory from the root (/) on down for a file named httpd.conf, and to print out the result of the search. Use find any time you need to locate a file. After locating httpd.conf, use an editor to put valid ServerAdmin and ServerName values into the configuration. In the Red Hat Linux 7.2 example, ServerAdmin is delivered with this default value:

ServerAdmin root@localhost

The e-mail address of root@localhost is a valid address, but it is not one we would want to advertise to remote users. We change ServerAdmin to

ServerAdmin webmaster@www.foobirds.org

The DocumentRoot directive points the server to the directory that contains web page information. By default, the Red Hat server gets web pages from the /var/www/html directory, as you can see by checking the value for DocumentRoot in the httpd.conf file:

```
$ grep '^DocumentRoot' httpd.conf
DocumentRoot "/var/www/html"
$ ls /var/www/html
index.html manual poweredby.png
```

You can put your own index.html file in this directory, along with any other supporting files and directories you need, and Apache will start serving out your data. Alternatively, you can edit the httpd.conf file to change the value in the DocumentRoot directive to point to the directory in which you store your data. The choice is yours. Either way, you need to create HyperText Markup Language (HTML) documents for the web server to display.

After the minimal changes are made to the httpd.conf file, the server can be restarted. The easiest way to do this on a Red Hat system is to run the /etc/init.d/httpd script file with the argument restart.

The *httpd.conf* File

To master Apache, you need to understand the Apache configuration file. Traditionally, Apache was configured by three files:

- **httpd.conf** Defines configuration settings for the HTTP protocol and for the operation of the server. This includes defining what directory holds the configuration files.
- **srm.conf** Configures how server requests are managed. This includes defining where HTTP documents and scripts are stored.
- **access.conf** Defines access control for the server and the information it provides. The functions of the three files overlap, and any Apache configuration directive can be used in any of the configuration files. The traditional division of the files into server, data, and security functions

The httpd.conf file includes containers that limit the scope of the directives they contain. For example, to limit certain directives to a specific directory, create a Directory container for those directives. Five different types of containers found in the Red Hat httpd.conf are:

- **<Directory *pathname*>** The Directory directive creates a container for directives that apply to the directory identified by *pathname*. Any configuration directives that occur after the Directory directive and before the next </Directory> statement apply only to the specified directory.
- **<Files *filename*>** The Files directive creates a container for directives that apply to the file identified by *filename*. Any configuration directives that occur after the Files directive and before the next </Files> statement apply only to the specified file. *filename* can refer to more than one file because it can contain the wildcard characters * and ?. Additionally, if the Files directive is followed by an optional ~ (tilde), the *filename* field is interpreted as a regular expression.

- **<Location *document*>** The Location directive creates a container for directives that apply to a specific document. Any configuration directives that occur after the Location directive and before the next </Location> statement apply only to the specified document.
- **<IfDefine *argument*>** The IfDefine directive creates a container for directives that are applied to the configuration only if the specified argument exists on the httpd command line. For example, the line <IfDefine HAVE_SSL> marks the beginning of a container of directives that are used only if the string –DHAVE_SSL occurs on the httpd command line. The IfDefine container ends with the next </IfDefine> statement.
- **<IfModule *module*>** The IfModule directive creates a container for directives that are applied to the configuration only if the specified module is loaded. For example, the directives enclosed by <IfModule mod_userdir.c> and <\IfModule> are used only if the module mod_userdir is loaded.

Loading Dynamic Shared Objects

The two directives that appear most in the Red Hat httpd.conf file are LoadModule and AddModule. These two directives make up more than 75 of the 250 active lines in the httpd configuration file. All 75 of these lines configure the Dynamic Shared Object (DSO) modules used by the Apache web server. Some systems may have many modules compiled into the Apache daemon. Others, such as the Red Hat Linux 7.2 system, are delivered with only two modules compiled in. These are:

- **http_core.c** This is the core module. It is always statically linked into the Apache kernel. It provides the basic functions that must be found in every Apache web server. This module is required. All other modules are optional.
- **mod_so.c** This module provides runtime support for Dynamic Shared Object modules. It is required if you plan to dynamically link in other modules at runtime. If modules are loaded through the httpd.conf file, this module must be installed in Apache to support those modules. For this reason, it is often statically linked into the Apache kernel.

DSO Modules Loaded in the Red Hat Configuration

Module	Purpose
mod_access	Specifies host- and domain-based access controls.
mod_actions	Maps a CGI script to a MIME file type.
mod_alias	Points to document directories outside the document tree.
mod_asis	Defines file types returned without headers.
mod_auth	Enables user authentication.
mod_auth_anon	Enables anonymous logins.
mod_auth_db	Enables use of a DB authentication file.
mod_autoindex	Enables automatic index generation.
mod_bandwidth	Sets bandwidth limits on server usage.
mod_cgi	Enables execution of CGI programs.
mod_dav	Provides WebDAV protocol extensions.
mod_dir	Controls formatting of directory listings.
mod_env	Allows CGI and SSI to inherit all shell environment variables.

Basic Server Directives

A few directives define basic information about the server itself. We modified two of these, ServerAdmin and ServerName, when creating the basic configuration. ServerAdmin defines the e-mail address of the web server administrator. In the default Red Hat configuration, this is set to root@localhost on the assumption that

there is always a root account, and there is always the hostname localhost. Change this to the full e-mail address of the real web administrator. For example:

```
ServerAdmin webmaster@www.foobirds.org
```

In this example, we use the classic web administratore – mailaddress webmaster@www.foobirds.org as the value for ServerAdmin. For this to work, we need a webmaster entry in the sendmail aliases file, which we created in Chapter 5, "Configuring a Mail Server"; and a CNAME record in the DNS database for www.foobirds.org.

ServerName defines the hostname returned to clients when they read data from this server. In the default Red Hat configuration, ServerName was commented out, and the example on the comment line set ServerName to localhost. Change this to provide a real hostname. For example:

```
ServerName www.foobirds.org
```

When the ServerName directive is commented out, the "real" hostname is sent to clients. Thus, if the name assigned to the first network interface is wren.foobirds.org, it is the name sent to clients when ServerName is undefined. Defining an explicit value for ServerName documents the configuration, and ensures that you get exactly the value you want. We set ServerName to www.foobirds.org so that, even though the web server is running on wren, the server will be known as www.foobirds.org during web interactions. Of course, www.foobirds.org must be a valid hostname configured in DNS.

The UseCanonicalName directive controls whether or not the value defined by ServerName is used. UseCanonicalName defines how httpd handles "self-referencing" URLs, which refer back to the server. When this is set to on, as it is in the Red Hat configuration, the value in ServerName is used. If it is set to off, the value that came in the query from the client is used. If your site uses multiple hostnames, you may want to set this to off so that the user will see the name they expect in the reply.

The ServerRoot option defines the directory that contains the httpd server files. This is different from DocumentRoot, which is the directory that contains the information files the server presents to clients. On Red Hat, and most other systems, this is /etc/httpd. The conf directory under the ServerRoot contains the three configuration files. Therefore, httpd.conf is itself located under the ServerRoot that it defines.

The ServerType option defines how the server is started. If the server starts from a startup script at boot time, the option is set to standalone. If the server is run on demand by inetd or xinetd, ServerType is set to inetd. Most of the time, web servers are in high demand, so it is best to start them at boot time. It is possible, however, for a user to set up a small, rarely used website on a Linux desktop. In that case, running the server from inetd or xinetd may be desirable. Port defines the TCP port number used by the server. The standard number is 80. On occasion, private web servers run on other port numbers. 8080 and 8000 are popular alternative ports for private websites. If you change the number, you must then tell your users the non-standard port number. For example, http://private.foobirds.org:8080 is a URL for a website running on TCP port 8080 on host private.foobirds.org.

When ServerType is set to inetd, it is usually desirable to set Port to something other than 80. The reason for this is that the ports under 1024 are "privileged" ports. If 80 is used, httpd must be run from inetd with the user ID root. This is a potential security problem, because an intruder might be able to exploit the website to get root access. Using port 80 is okay when ServerType is standalone because the initial httpd process does not provide direct client service. Instead, it starts several other HTTP daemons to provide client services that do not run with root privilege.

Defining Virtual Hosts

Virtual hosts allow a server to host multiple websites known by different hostnames. The Red Hat configuration has an entire section dedicated to virtual hosts, but it is all commented out. It is there only to serve as an example. To use virtual hosts, you must first uncomment the NameVirtualHost directive to enable name-based virtual hosts. There are IP-based virtual hosts, but those consume valuable IP addresses. Name-based virtual hosts are recommended by the Apache developers, and are preferred by most administrators.

On the sample Red Hat system, the NameVirtualHost directive is commented-out. The line is:

```
#NameVirtualHost *
```

The asterisk on this line stands for any address assigned to any interface on the host. To make this more understandable, we will be more explicit. We remove the hash mark (#) to activate the line, and set the NameVirtualHost address to the primary address of our server:

```
NameVirtualHost 172.16.5.1
```

Next, define the virtual hosts that will be served. For example, to host websites for fish.edu and mammals.com on the wren.foobirds.org server, add the following lines to the httpd.conf file:

```
<VirtualHost www.fish.edu>
       DocumentRoot /var/www/html/fish
       ServerName www.fish.edu
</VirtualHost>
<VirtualHost www.mammals.com>
       DocumentRoot /var/www/html/mammals
       ServerName www.mammals.com
</VirtualHost>
```

Each VirtualHost directive defines a hostname alias to which the server responds. For this to be valid, DNS must define the alias with a CNAME record. The example requires CNAME records that assign wren.foobirds.org the aliases of www.fish.edu and www.mammals.com. When wren receives a server request addressed to one of these aliases, it uses the configuration parameters defined here to override its normal settings. Therefore, when it gets a request for www.fish.edu, it uses www.fish.edu as its ServerName value instead of its own server name, and uses /var/www/ html/fish as the DocumentRoot.

Setting Up an FTP Server

File Transfer Protocol (FTP) has been the standard method for sharing files over the Internet for many years. Even with the popularity of the Web, which made document database services such as Gopher and WAIS almost obsolete, FTP servers are still the most common way to make directories of documents and software available to the public over the Internet.

File-sharing applications, such as NFS and Samba, are excellent tools for sharing files and directories over a private network. For organizations that need to share large numbers of files over public networks, however, FTP server software provides more robust tools for sharing files and protecting your computer systems. Also, FTP client software (for accessing FTP servers) is available for any type of computer that can access a network.

Attributes of FTP servers

That FTP was implemented on large, multiuser UNIX systems accounts for many of the design decisions that remain a part of FTP today. As the most popular free version of UNIX, Linux in general (and Red Hat Linux in particular) has drawn on FTP features that have resulted from years of testing and experience gained from other UNIX versions of FTP. Some attributes of FTP servers follow:

- Because FTP was originally used on multiuser systems, only restricted parts of the file system in Red Hat Linux are devoted to FTP. Those who access FTP from a public user account (usually the anonymous user name) are automatically given an FTP directory (often /var/ftp) as their root directory. From there, the anonymous user can access only files and directories below that point in the file system.
- Access to the FTP server relies on a login process that uses standard UNIX login names (that is, those user names found in /etc/passwd). Although anonymous was the user name that strangers would use, users with their own accounts on the system could log in with their own user names through FTP and most likely have access to a greater part of the file system (in particular, their own private files and directories).
- The ftp command and other FTP client programs let you log in and then operate from a command interpreter (similar to a very simple shell). Many of the commands that you use from that command interpreter are familiar UNIX commands. You change directories with cd, list files with ls, change permissions with chmod,

and check your location with pwd (to name a few). When you find whereyou want to be, you use the get command to download a file or the put command to upload one.

As an administrator of an FTP server, it is your responsibility to make sure that you share your files in a way that gives people access to the information that you want them to have without compromising the security of your system. This means implementing a strong security policy and relentlessly monitoring the system to prevent abuse.

Running the FTP Server

When you install Red Hat Linux, your system is already set up as an FTP server. However, although users can log in and see the default FTP directories, no files that they can access are there yet. The default setup for your Red Hat Linux FTP server after you install the FTP software follows (the Washington University FTP Server software is the wu-ftpd package)

- **FTP Daemon** — The FTP daemon is set up in the /etc/xinetd.d/wu-ftpd file as /usr/sbin/in.ftpd. When someone requests FTP service from your computer (probably on port 21), the xinetd daemon (which listens to lots of ports) starts the FTP daemon to handle that login request. The daemon runs with the -l option (so that FTP requests are logged by syslog to /var/log/messages) and the -a option (so that the /etc/ftpaccess file is used to define access permissions).
- **Access Permissions** (/etc/ftpaccess) — The ftpaccess file delivered with Linux is quite restrictive. Although you can change it to be as open or as restrictive as you like, here is how the file is originally set:
    - **deny** and **allow** — The following lines are included to set which user accounts are allowed and which are denied access to the FTP service:

```
deny-uid %-99 %65534-
deny-gid %-99 %65534-
allow-uid ftp
allow-gid ftp
```

      The deny-uid and deny-gid entries prevent access to the FTP service from any users with IDs that are 99 or less or 65534 or greater for either user or group accounts. The allow-uid and allow-gid lines make an exception to the deny rules by allowing the ftp user and group to use the FTP service.
    - **email root@localhost** — E-mail related to the administration of the FTP server is directed to the root user on the local computer, by default.
    - **loginfails 5** — The FTP connection terminates after five consecutive failed login attempts. (This slows down people who are trying to guess your server's passwords.)
    - **readme README*** — When the user logs in (login) or changes to any other accessible directory (cwd=•), the user is notified of the existence of README files, if they exist. By default, none of these files exist. For any file that begins with the word README, a message is displayed by the server that says "Please read the file README.whatever."
    - **message** — This indicates that the message contained in the /welcome.msg file should be displayed when a user logs in to FTP. A similar line indicates that the .message file is displayed when the user enters a directory that contains such a file. By default, none of these files exist. If you want them on your FTP server, you have to create them yourself.
    - **compress yes all** — This enables compression of files for the FTP site for all users. The compress command is the standard compression command used in UNIX systems (though gzip is used more often with free operating systems, such as Linux). The compress command lines used to carry out the compressions are defined in /etc/ftpconversions. (Files stored by compress have a .Z suffix.)
    - **tar yes all** — This enables tar compression for all users at the FTP site. The tar command is the standard UNIX command used to create archives of multiple files. The tar commands used to carry out the compressions are defined in /etc/ftpconversions. (Files stored by tar have a .tar, .tar.gz, or .tar.Z suffix.)
    - **chmod no guest, anonymous** — This prevents guest and anonymous user names from changing the permissions on any files or directories.
    - **delete no anonymous** — This prevents anonymous user name from deleting files or directories.
    - **overwrite no anonymous** — This prevents anonymous user name from overwriting existing files or directories.
    - **rename no anonymous** — This prevents anonymous user name from renaming any files or directories.

- **log transfers anonymous, guest, real inbound, outbound** — This logs file transfers for the anonymous user, guest user, and any real users (that is, those who have their own user accounts on the Linux system). Both uploads (inbound) and downloads (outbound) transfers are logged.
- **shutdown /etc/shutmsg** — This checks the /etc/shutmsg file to see if the server is about to be shut down. If it is, your FTP server sends a message to current FTP users, warning them that the server is about to go down. It also denies new FTP connections and disconnects current users at a specified time prior to shutdown. (By default, the /etc/shutmsg file does not exist.)
- **passwd-check rfc822 warn** — This checks that passwords for anonymous logins are rfc822-compliant addresses. In other words, the FTP server asks for any valid e-mail address as the password for the anonymous login. If the address is not compliant (that is, is not in the form user@host.domain), the server will "warn" the user but still allow the user to log in.

- **FTP Root Directory** — For a user who logs in as an anonymous user, the /var/ftp directory is assigned as the user's root directory. In other words, the anonymous user could not cd above the /var/ftp directory (or even know what files exist outside the /var/ftp directory structure).

  Within the /var/ftp directory are those directories and files necessary to make FTP work properly, without your having to access other files in the file system. The /bin directory contains executable commands that FTP may need (such as compress, ls, and gzip). The /etc directory contains passwd and group entries. The /lib directory holds shared object libraries needed by FTP. Finally, the /pub directory is available for placing the files that you want to be generally available to anonymous users.

Shutting Down and Restarting the FTP Server

With ftpshut command, you can either shut down the server immediately or set it to shut down some time in the future. When you run ftpshut, it creates a shutdown file. To restart the FTP service after it has been shut down, you need to remove that file.

Shutting down FTP

The ftpshut command shuts down the FTP service. Any users who are currently connected to the FTP service are disconnected. When they type FTP commands, they will see a message such as FTP server shut down — please try again later or simply Not connected. To shut down the service immediately, type the following:

```
ftpshut now
```

You can set the server to shut down at a specific time by either adding the number of minutes from now that you want the server to shut down or by entering a particular time in the future. Here are some examples:

```
ftpshut 20:18
ftpshut -l 20 -d 3
```

In the first example, the server is set to shut down at 20:18 (or 8:18 p.m. on a 12-hour clock). In the second example, the -l option specifies the number of minutes left before the server will be shut down (in this example, 20 minutes). The -d option specifies the number of minutes before the FTP service shuts down that users are disconnected from the server. (If no -d option is specified, users are disconnected five minutes To be less antisocial about shutting down your FTP service, it's nice to send a warning message to let users know that FTP is going down. Here's an example of how to send a shutdown-warning message when you shut down your FTP server:

```
ftpshut 20:15 "FTP will go down soon. Finish and disconnect."
```

Restarting FTP

After ftpshut is run, it creates a file called shutmsg in both the /etc and /var/ftp/etc directories. When your FTP server shuts down, it stays down until these files are removed. So, to restart your FTP server, all you have to do is the following (as root user):

```
rm /etc/shutmsg /var/ftp/etc/shutmsg
```

The above command removes the files. At this point, users will immediately be able to reconnect to your FTP server. Another command you could use to restart your FTP server is the ftprestart command.