Module 19: System Administration in UNIX

In the context of the OS service provisioning, system administration plays a pivotal role. This is particularly the case when a system is accessed by multiple users. The primary task of a system administrator is to ensure that the following happens:

a. The top management is assured of efficiency in utilization of the system's resources.
b. The general user community gets the services which they are seeking.

In other words, system administrators ensure that there is very little to complain about the system's performance or service availability.

In Linux environment with single user PC usage, the user also doubles up as a system administrator. Much of what we discuss in Unix context applies to Linux as well.

In all Unix flavours there is a notion of a superuser privilege. Most major administrative tasks require that the system administrator operates in the superuser mode with root privileges. These tasks include starting up and shutting down a system, opening an account for a new user and giving him a proper working set-up. Administration tasks also involve installation of new software, distributing user disk space, taking regular back-ups, keeping system logs, ensuring secure operations and providing network services and web access.

We shall begin this module by enlisting the tasks in system administration and offering exposition on most of these tasks as the chapter develops.

19.1 Unix Administration Tasks

Most users are primarily interested in just running a set of basic applications for their professional needs. Often they cannot afford to keep track of new software releases and patches that get announced. Also, rarely they can install these themselves. In addition, these are non-trivial tasks and can only be done with superuser privileges.

Users share resources like disk space, etc. So there has to be some allocation policy of the disk space. A system administrator needs to implement such a policy. System administration also helps in setting up user's working environments.

On the other hand, the management is usually keen to ensure that the resources are used properly and efficiently. They seek to monitor the usage and keep an account of system usage. In fact, the system usage pattern is often analysed to help determine the efficacy of

usage. Clearly, managements' main concerns include performance and utilisation of resources to ensure that operations of the organisation do not suffer.

At this juncture it may be worth our while to list major tasks which are performed by system administrators. We should note that most of the tasks require that the system administrator operates in superuser mode with root privileges.

19.1.1 Administration Tasks List

This is not an exhaustive list, yet it represents most of the tasks which system administrators perform:

1. System startup and shutdown: In the Section 19.2, we shall see the basic steps required to start and to stop operations in a Unix operational environment.
2. Opening and closing user accounts: In Unix an administrator is both a user and a super-user. Usually, an administrator has to switch to the super-user mode with root privileges to open or close user accounts. In Section 19.3, we shall discuss some of the nuances involved in this activity.
3. Helping users to set up their working environment: Unix allows any user to customize his working environment. This is usually achieved by using .rc files. Many users need help with an initial set-up of their .rc files. Later, a user may modify his .rc files to suit his requirements. In Section 19.4, we shall see most of the useful .rc files and the interpretations for various settings in these files.
4. Maintaining user services: Users require services for printing, mail Web access and chat. We shall deal with mail and chat in Section 19.4 where we discuss .rc files and with print services in Section 19.5 where we discuss device management and services. These services include spooling of print jobs, provisioning of print quota, etc.
5. Allocating disk space and re-allocating quotas when the needs grow: Usually there would be a default allocation. However, in some cases it may be imperative to enhance the allocation. We shall deal with the device oriented services and management issues in Section 19.5.
6. Installing and maintaining software: This may require installing software patches from time to time. Most OSs are released with some bugs still present. Often with usage these bugs are identified and patches released. Also, one may have some software installed which satisfies a few of the specialized needs of the user

community. As a convention this is installed in the directory /usr/local/bin. The local is an indicator of the local (and therefore a non-standard) nature of software. We shall not discuss the software installation as much of it is learned from experienced system administrators by assisting them in the task.

7. Installing new devices and upgrading the configuration: As a demand on a system grows, additional devices may need to be installed. The system administrator will have to edit configuration files to identify these devices. Some related issues shall be covered in section 19.5 later in this chapter.
8. Provisioning the mail and internet services: Users connected to any host shall seek Mail and internet Web access. In addition, almost every machine shall be a resource within a local area network. So for resource too the machine shall have an IP address. In most cases it would be accessible from other machine as well. We shall show the use .mailrc files in this context later in Section 19.4.
9. Ensuring security of the system: The internet makes the task of system administration both interesting and challenging. The administrators need to keep a check on spoofing and misuse. We have discussed security in some detail in the module on OS and Security.
10. Maintaining system logs and profiling the users: A system administrator is required to often determine the usage of resources. This is achieved by analysing system logs. The system logs also help to profile the users. In fact, user profiling helps in identifying security breaches as was explained in the module entitled OS and Security.
11. System accounting: This is usually of interest to the management. Also, it helps system administrators to tune up an operating system to meet the user requirements. This also involves maintaining and analysing logs of the system operation.
12. Reconfiguring the kernel whenever required: Sometimes when new patches are installed or a new release of the OS is received, then it is imperative to compile the kernel. Linux users often need to do this as new releases and extensions become available.

Let us begin our discussions with the initiation of the operations and shutdown procedures.

19.2 Starting and Shutting Down

First we shall examine what exactly happens when the system is powered on. Later, we shall examine the shutdown procedure for Unix systems. Unix systems, on being powered on, usually require that a choice be made to operate either in single or in multiple-user mode. Most systems operate in multi-user mode. However, system administrators use single-user mode when they have some serious reconfiguration or installation task to perform. Family of Unix systems emanating from System V usually operate with a *run level*. The single-user mode is identified with run level *s*, otherwise there are levels from 0 to 6. The run level 3 is the most common for multi-user mode of operation.

On being powered on, Unix usually initiates the following sequence of tasks:

1. The Unix performs a sequence of self-tests to determine if there are any hardware problems.
2. The Unix kernel gets loaded from a root device.
3. The kernel runs and initializes itself.
4. The kernel starts the init process. All subsequent processes are spawned from init process.
5. The init checks out the file system using fsck.
6. The init process executes a system boot script.
7. The init process spawns a process to check all the terminals from which the system may be accessed. This is done by checking the terminals defined under /etc/ttytab or a corresponding file. For each terminal a getty process is launched. This reconciles communication characteristics like baud rate and type for each terminal.
8. The *getty* process initiates a login process to enable a prospective login from a terminal.

During the startup we notice that *fsck* checks out the integrity of the file system. In case the *fsck* throws up messages of some problems, the system administrator has to work around to ensure that there is a working configuration made available to the users. It will suffice here to mention that one may monitor disk usage and reconcile the disk integrity.

The starting up of systems is a routine activity. The most important thing to note is that on booting, or following a startup, all the temporary files under *tmp* directory are cleaned

up. Also, zombies are cleaned up. System administrators resort to booting when there are a number of zombies and often a considerable disk space is blocked in the *tmp* directory.

We next examine the shutdown. Most Unix systems require invoking the shutdown utility. The shutdown utility offers options to either halt immediately, or shutdown after a pre-assigned period. Usually system administrators choose to shutdown with a pre-assigned period. Such a shutdown results in sending a message to all the terminals that the system shall be going down after a certain interval of time, say 5 minutes. This cautions all the users and gives them enough time to close their files and terminate their active processes. Yet another shutdown option is to reboot with obvious implications. The most commonly used shutdown command is as follows:

shutdown -h time [message]

Here the time is the period and message is optional, but often it is intended to advise users to take precautions to terminate their activity gracefully. This mode also prepares to turn power off after a proper shutdown. There are other options like k, r, n etc. The readers are encouraged to find details about these in Unix man pages. For now, we shall move on to discuss the user accounts management and run command files.

19.3 Managing User Accounts

When a new person joins an organisation he is usually given an account by the system administrator. This is the login account of the user. Now a days almost all Unix systems support an admin tool which seeks the following information from the system administrator to open a new account:

1. Username: This serves as the login name for the user.
2. Password: Usually a system administrator gives a simple password. The users are advised to later select a password which they feel comfortable using. User's password appears in the shadow files in encrypted forms. Usually, the /etc/passwd file contains the information required by the login program to authenticate the login name and to initiate appropriate shell as shown in the description below:

   bhatt:x:1007:1::/export/home/bhatt:/usr/local/bin/bash

   damu:x:1001:10::/export/home/damu:/usr/local/bin/bash

   Each line above contains information about one user. The first field is the name of the user; the next a dummy indicator of password, which is in another file, a shadow file. Password programs use a trap-door algorithm for encryption.

3. Home directory: Every new user has a home directory defined for him. This is the default login directory. Usually it is defined in the run command files.
4. Working set-up: The system administrators prepare .login and .profile files to help users to obtain an initial set-up for login. The administrator may prepare .cshrc, .xinitrc .mailrc .ircrc files. In Section 19.4 we shall later see how these files may be helpful in customizing a user's working environment. A natural point of curiosity would be: what happens when users log out? Unix systems receive signals when users log out. Recall, in Section 19.2 we mentioned that a user logs in under a login process initiated by *getty* process. Process *getty* identifies the terminal being used. So when a user logs out, the *getty* process which was running to communicate with that terminal is first killed. A new *getty* process is now launched to enable yet another prospective login from that terminal.

   The working set-up is completely determined by the startup files. These are basically .rc (run command) files. These files help to customize the user's working environment. For instance, a user's .cshrc file shall have a path variable which defines the access to various Unix built-in shell commands, utilities, libraries etc. In fact, many other shell environmental variables like HOME, SHELL, MAIL, TZ (the time zone) are set up automatically. In addition, the .rc files define the access to network services or some need-based access to certain licensed software or databases as well. To that extent the *.rc* files help to customize the user's working environment.

   We shall discuss the role of run command files later in Section 19.4.
5. Group-id: The user login name is the user-id. Under Unix the access privileges are determined by the group a user belongs to. So a user is assigned a group-id. It is possible to obtain the id information by using an id command as shown below:

   ```
   [bhatt@iiitbsun OS]$id
   uid=1007(bhatt) gid=1(other)
   [bhatt@iiitbsun OS]$
   ```
6. Disc quota: Usually a certain amount of disk space is allocated by default. In cases where the situation so warrants, a user may seek additional disk space. A user may interrogate the disk space available at any time by using the *df* command. Its usage is shown below:

df [options] [name] : to know the free disk space.

where name refers to a mounted file system, local or remote. We may specify directory if we need to know the information about that directory. The following options may help with additional information:

-l : for local file system

-t : reports total no. of allocated blocks and i-nodes on the device.

The Unix command *du* reports the number of disk blocks occupied by a file. Its usage is shown below:

du [options] [name]... where name is a directory or a file

Above name by default refers to the current directory. The following options may help with additional information:

-a : produce output line for each file

-s : report only the total usage for each name that is a directory i.e. not individual files.

-r : produce messages for files that cannot be read or opened

7. Network services: Usually a user shall get a mail account. We will discuss the role of .mailrc file in this context in section 19.4. The user gets an access to Web services too.
8. Default terminal settings: Usually vt100 is the default terminal setting. One can attempt alternate terminal settings using *tset, stty, tput,* tabs with the control sequences defined in *terminfo termcap* with details recorded in /etc/ttytype or /etc/tty files and in shell variable TERM. Many of these details are discussed in Section 19.5.1 which specifically deals with terminal settings. The reader is encouraged to look up that section for details.

Once an account has been opened the user may do the following:

1. Change the pass-word for access to one of his liking.
2. Customize many of the run command files to suit his needs.

Closing a user account: Here again the password file plays a role. Recall in section 19.1 we saw that /etc/password file has all the information about the users' home directory, password, shell, user and group-id, etc. When a user's account is to be deleted, all of this information needs to be erased. System administrators login as root and delete the user entry from the password file to delete the account.

19.4 The *.rc* Files

Usually system administration offers a set of start-up run command files to a new user. These are files that appear as .rc files. These may be *.profile, .login, .cshrc, .bashrc .xinitrc, .mailrc .ircrc,* etc. The choice depends upon the nature of the login shell. Typical allocations may be as follows:

0 Bourne or Korn shell: *.profile*

1 C-Shell: *.login, .cshrc*

2 BASH: *.bashrci*

3 TCSH: *.tcshrc*

BASH is referred as Bourne-again shell. TCSH is an advanced C-Shell with many shortcuts like pressing a tab may complete a partial string to the extent it can be covered unambiguously. For us it is important to understand what is it that these files facilitate.

Role of *.login* and *.profile* files: The basic role of these files is to set up the *environment* for a user. These may include the following set-ups.

- Set up the terminal characteristics: Usually, the set up may include terminal type, and character settings for the prompt, erase, etc.
- Set up editors: It may set up a default editor or some specific editor like emacs.
- Set up protection mode: This file may set up umask, which stands for the user mask. umask determines access right to files.
- Set up environment variables: This file may set up the path variable. The path variable defines the sequence in which directories are searched for locating the commands and utilities of the operating system.
- Set up some customization variables: Usually, these help to limit things like selecting icons for mail or core dump size up to a maximum value. It may be used for setting up the limit on the scope of the command history, or some other preferences.

A typical *.login* file may have the following entries:

```
# A typical .login file
umask 022
setenv PATH /usr/ucb:/usr/bin:/usr/sbin:/usr/local/bin
setenv PRINTER labprinter
```

```
setenv EDITOR vi
biff y
set prompt='hostname'=>
```

The meanings of the lines above should be obvious from the explanation we advanced earlier. Next we describe *.cshrc* files and the readers should note the commonalities between these definitions of initialisation files.

The *.cshrc* file: The C-shell makes a few features available over the Bourne shell. For instance, it is common to define aliases in *.cshrc* files for very frequently used commands like *gh* for *ghostview* and *c* for *clear*. Below we give some typical entries for *.cshrc* file in addition to the many we saw in the *.login* file in this section:

```
if (! $?TERM) setenv TERM unknown
if ("TERM" == "unknown" || "$TERM" == "network") then echo -n 'TERM? [vt100]: ';
set ttype=($<)
if (ttype == "") set ttype="vt100"
if (ttype == "pc") then set ttype="vt100"
endif
setenv TERM $ttype
endif
alias cl clear
alias gh ghostview
set history = 50
set nobeep
```

Note that the above, in the first few lines in the script, system identifies the nature of terminal and sets it to operate as vt100. It is highly recommended that the reader should examine and walk-through the initialization scripts which the system administration provides. Also, a customization of these files entails that as a user we must look up these files and modify them to suit our needs.

There are two more files of interest. One corresponds to regulating the mail and the other which controls the screen display. These are respectively initialized through *.mailrc* and *.xinitrc*. We discussed the latter in the chapter on X Windows. We shall discuss the settings in *.mailrc* file in the context of the mail system.

The mail system: *.mailrc* file : From the viewpoint of the user's host machine, the mail program truly acts as the main anchor for our internet-based communication. The Unix sendmail program together with the uu class of programs form the very basis of the mail under Unix. Essentially, the mail system has the following characteristics:

1. The mail system is a *Store and forward* system.
2. Mail is picked up from the mail server periodically. The *mail* daemon, picks up the mail running as a background process.
3. Mail is sent by *sendmail* program under Unix.
4. The *uu* class of programs like *uucp* or Unix-to-Unix copy have provided the basis for developing the mail tools. In fact, the file attachments facility is an example of it.

On a Unix system it is possible to invoke the mail program from an auto-login or .cshrc program.

Every Unix user has a mailbox entry in the /usr/spool/mail directory. Each person's mail box is named after his own username. In Table 19.1 we briefly review some very useful mail commands and the wild card used with these commands.

We next give some very useful commands which help users to manage their mails efficiently:

Command	Interpretation
?	means all the mails
+	means the next mail
-	means the previous mail
!	escape to shell (usually to compose and edit)
sh	escape to shell (an alternative to using !)
top	shows first few lines of the message
Wild cards	**Interpretations**
*	for all
.	for current
$	for last
^	for first
:r	for messages that have been read
:u	for as yet unread
/pattern/	for messages with pattern

Table 19.1: Various command options for mail.

d:r : delete all read messages.

d:usenet : delete all messages with usenet in body

p:r : print all read messages.

p:bhatt : print all from user ``bhatt".

During the time a user is composing a mail, the mail system tools usually offer facility to escape to a shell. This can be very useful when large files need to be edited along side the mail being sent. These use ~ commands with the interpretations shown below:

~! escape to shell,

~d include dead.letter

~h edit header field

The mail system provides for command line interface to facilitate mail operations using some of the following commands. For instance, every user has a default mail box called mbox. If one wishes to give a different name to the mailbox, he may choose a new name for it. Other facilities allow a mail to be composed with, or without, a subject or to see the progress of the mail as it gets processed. We show some of these options and their usage with mail command below.

mail -s greetings user@machine.domain

-s: option is used to send a mail with subject.

-v: option is for the verbose option, it shows mails' progress

-f mailbox: option allows user to name a new mail box

mail -f newm: where newm may be the new mail box option which

a user may opt for in place of mbox (default option).

Next we describe some of the options that often appear inside *.mailrc* user files. Generally, with these options we may have aliases (nick-names) in place of the full mail address. One may also set or unset some flags as shown in the example below:

unset askcc

set verbose

set append

Command	Interpretation
append	Appends all messages in mbox as opposed to pre-pending them.
asks	Asks for subject.
askcc	Prompts for carbon copy.
autoprint	When set equal to `dp` it deletes and prints the next message.
.	The dot symbol, when set, terminates mail with a . on column 1.
ignore	When set, ignores control_c depression.
metoo	When set, always generates a copy for the sender.
nosave	When set, it does not save an aborted message in dead.letter
replyall	Reverses the sense of `r` and `R` i.e. send replies to all in CC besides the sender.
verbose	When set, it is like `-v` option in the mail command.
	The PAGER variable can be used to specify a tool to use in order to slow down the output of a message. The *more* program may be used. *more* slows down the text output to one screen at a time.

Table 19.2: Various options for .mailrc file.

In Table 19.2, we offer a brief explanation of the options which may be set initially in .mailrc files.

In addition, in using the mail system the following may be the additional facilities which could be utilized:

1. To subscribe to listserv@machine.domain, the body of the message should contain “subscribe", the group to subscribe to and the subscribers' e-mail address as shown in the following example.

   subscribe allmusic me@mymachine.mydomain.

2. To unsubscribe use logout allmusic. In addition to the above there are vacation programs which send mails automatically when the receiver is on vacation.

   Mails may also be encrypted. For instance, one may use a pretty good privacy (PGP) for encrypting mails.

Facilitating chat with *.ircrc* **file:** System administrators may prepare terminals and offer Inter Relay Chat or IRC facility as well. IRC enables real-time conversation with one or more persons who may be scattered anywhere globally. IRC is a multi-user system. To use IRC's, Unix-based IRC versions, one may have to set the terminal emulation to vt100 either from the keyboard or from an auto-login file such as .login in bin/sh or .cshrc in /bin/csh.

```
$ set TERM=vt100
$ stty erase "^h"
```

The most common way to use the IRC system is to make a telnet call to the IRC server. There are many IRC servers. Some servers require specification of a port number as in irc.ibmpcug.co.uk9999.

When one first accesses the IRC server, many channels are presented. A channel may be taken as a discussion area and one may choose a channel for an online chat (like switch a channel on TV). IRCs require setting up an .ircrc file. Below we give some sample entries for a .ircrc file. The .ircrc files may also set internal variables.

/COMMENT .....

/NICK <nn>

/JOIN <ch>

IRC commands begin with a \/" character. In Table 19.3, we give a few of the commands for IRC with their interpretations.

Command	Interpretation
/HELP	to seek help
/QUIT /EXIT /BYE /SIGNOFF	to quit
/LIST	to list topics of discussion
/JOIN	to join
/NICK	to change nick name
/MSG	to send message
/QUERY	to send a message
/WHOIS	to identify a user
/AWAY	to indicate if you are going to be away
/AWAY or /QUERY	without arguments to deactivate the mode
/MOTD	for message of the day on a server
/ADMIN	to get administrative information
/USERS /LUSERS	to get users on servers
/TIME	server's local time
/INVITE	to invite a user to join a discussion group
/msg , < .. >	, is in response to the last message received
/msg . < ... >	. is for continuation with the last message sent
/LASTLOG	to see IRC log information
/NOTICE < *nn|channel* >< *msg* >	to send a private message
/IGNORE	to ignore all contact from a user or all users
/IGNORE < *nn* > | < *user@host* > [−] < *message − type* >	
/KICK [channel] < *nn* >	to, kick nn from channel
/ME	like whoami, gives the channel and tells what you are doing

Table 19.3: Various commands with interpretation.

IRCs usually support a range of channels. Listed below are a few of the channel types:

Limbo or Null

Public

Private

Secret

Moderated

Limited

Topic limited

Invite Only

Message disabled.

The above channel types are realized by using a mode command. The modes are set or unset as follows. The options have the interpretations shown in Table 19.4.

/MODE sets (with +) and unsets (with -) the mode of channel with the following options

/MODE <channel> +<channel options> < parameters>

/MODE <channel> -<channel options> < parameters>

Option	Interpretation
i	invite only channel
l < *n* >	make channel limited to maximum of *n* users
m	make a channel moderated
n	no messages to the channel are permitted
o < *nn* >	Makes a person a channel operator
s	makes a channel secret
t	makes a channel topic limited

Table 19.4: Various options for channels.

19.4.1 Sourcing Files

As we have described above, the .rc files help to provide adequate support for a variety of services. Suppose we are logged to a system and seek a service that requires a change in one of the *.rc* files. We may edit the corresponding file. However, to affect the changed behavior we must source the file. Basically, we need to execute the source command with the file name as argument as shown below where we source the *.cshrc* file:

source .cshrc

19.5 Device Management and Services

Technically the system administrator is responsible for every device, for all of its usage and operation. In particular, the administrator looks after its installation, upgrade, configuration, scheduling, and allocating quotas to service the user community. We shall, however, restrict ourselves to the following three services:

1. Terminal-based services, discussed in Section 19.5.1
2. Printer services, discussed in Section 19.5.2
3. Disc space and file services, discussed in Section 19.5.3.

We shall begin with the terminal settings and related issues.

19.5.1 The Terminal Settings

In the context of terminal settings the following three things are important:

1. Unix recognizes terminals as special files.
2. Terminals operate on serial lines. Unix has a way to deal with files that are essentially using serial communication lines.
3. The terminals have a variety of settings available. This is so even while the protocols of communication for all of them are similar.

From the point of terminal services provisioning and system configuration, system administration must bear the above three factors in mind. Unix maintains all terminal related information in *tty* files in /etc/dev directory. These files are special files which adhere to the protocols of communication with serial lines. This includes those terminals that use modems for communication. Some systems may have a special file for console like /etc/dev/console which can be monitored for messages as explained in the chapter on X-Windows. Depending upon the terminal type a serial line control protocol is used which can interrogate or activate appropriate pins on the hardware interface plug.

The following brief session shows how a terminal may be identified on a host:

login: bhatt

Password:

Last login: Tue Nov 5 00:25:21 from 203.197.175.174

[bhatt@iiitbsun bhatt]$hostname

iiitbsun

[bhatt@iiitbsun bhatt]$tty

/dev/pts/1

[bhatt@iiitbsun bhatt]$

termcap **and** *terminfo* **files:** The *termcap* and *terminfo* files in the directory /etc or in */usr/share/lib/terminfo* provide the terminal database, information and programs for use in the Unix environment. The database includes programs that may have been compiled to elicit services from a specific terminal which may be installed. The programs that control the usage of a specific terminal are identified in the environment variable TERM as shown in the example below:

[bhatt@localhost DFT02]$ echo $TERM

xterm

[bhatt@localhost DFT02]$

option	value	Sample Value
n	Baud rate	9600
rows n	Number of rows on the screen of the terminal	24 to 36
columns n	Number of columns on the screen of the terminal	80
erase c	Erase character	control h
eof c	End of file character	control d
intr c	Interrupt character	control c
oddp	Enable odd parity	oddp
-cstopb	Use one stop bit (instead of two)	-cstopb

Table 19.5: Options under *stty*.

There are specific commands like tic, short for terminal information compilation. Also, there are programs that convert *termcap* to *terminfo* whenever required. For detailed discussions on terminal characteristics and how to exploit various features the reader may refer to [2]. We shall, however, elaborate on two specific commands here.

These are the tset and stty commands.

1. ***tset* Command**: The tset command is used to initialize a terminal. Usually, the command sets up initial settings for characters like erase, kill, etc. Below we show how under C-Shell one may use the tset command:

   $setenv TERM `tset - Q -m ":?vt100"

   Sometimes one may prepare a temporary file and source it.

2. ***stty* command**: We briefly encountered the *stty* command in Section 19.2. Here we shall elaborate on *stty* command in the context of options and the values which may be availed by using the *stty* command. In Table 19.5 we list a few of the options with their corresponding values.

   There are many other options. In Table 19.5 we have a sample of those that are available. Try the command *stty* -a to see the options for your terminal. Below is shown the setting on my terminal:

   [bhatt@localhost DFT02]$ stty -a

   speed 38400 baud; rows 24; columns 80; line = 0;

   intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = M-^?; eol2 = M-^?;

   start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V;

   flush = ^O; min = 1; time = 0;

   -parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts

   -ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff

   -iuclc ixany imaxbel

opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
[bhatt@localhost DFT02]$

Lastly, we discuss how to attach a new terminal. Basically we need to connect a terminal and then we set-up the entries in *termcap* and/or in *terminfo* and configuration files. Sometimes one may have to look at the /etc/inittab or /etc/ttydefs as well. It helps to reboot the system on some occasions to ensure proper initialization following a set-up attempt.

19.5.2 Printer Services

Users obtain print services through a printer daemon. The system arranges to offer print services by spooling print jobs in a spooling directory. It also has a mechanism to service the print requests from the spooling directory. In addition, system administrators need to be familiar with commands which help in monitoring the printer usage. We shall begin with a description of the *printcap* file.

The printcap file: Unix systems have their print services offered using a spooling system. The spooling system recognizes print devices that are identified in */etc/printcap* file. The *printcap* file serves not only as a database, but also as a configuration file. Below we see the *printcap* file on my machine:

```
# /etc/printcap
#
# DO NOT EDIT! MANUAL CHANGES WILL BE LOST!
# This file is autogenerated by printconf-backend during lpd init.
#
# Hand edited changes can be put in /etc/printcap.local, and will be included.
iiitb:\
    :sh:\
    :ml=0:\
    :mx=0:\
    :sd=/var/spool/lpd/iiitb:\
    :lp=|/usr/share/printconf/jetdirectprint:\
    :lpd_bounce=true:\
```

Print command	Interpretation
lpr	Unix print command. May choose a printer by default or require user to give the printer name
lpq	Gives the status of print queue. Sometimes it helps to determine your position in the print queue
lprm	In case the print request is to be removed before the print daemon picks it up for printing.
lpc	It is a class of administrative commands. Below we show administrative information which is obtained by using these sub-commands. Note each of these use a printer as an argument to specify the pritner to which the reference is made
Sub-command	Interpretation All these commands require to specify a printer as an argument. Example: status myprinter
status	Shows the queue on a specified printer Also, gives other relevant status information
start	Starts print service on the specified printer
abort	Aborts print service on the specified printer
stop	Stop print service on specified printer
enable disable	Enables or disables the specified printer

:if=/usr/share/printconf/mf_wrapper:

The *printcap* file is a read-only file except that it can be edited by superuser ROOT.

The entries in *printcap* files can be explained using Table 19.6. With the file description and the table we can see that the spooling directory for our printer, with printer name iiitb is at /var/spool. Also note we have no limit on file size which can be printed.

Table 19.6: The *printcap* file: printer characteristics.

Printer spooling directory: As we explained earlier, print requests get spooled first. Subsequently, the printer daemon *lpd* honours the print request to print. To achieve this, one may employ a two layered design. Viewing it bottom up, at the bottom layer maintain a separate spooling directory for each of the printers. So, when we attach a new printer, we must create a new spooling directory for it. At the top level, we have a spooling process which receives each print request and finally spools it for printer(s). Note that the owner of the spool process is a group *daemon.*

Printer monitoring commands: The printer commands help to monitor both the health of the services as also the work in progress. In table 19.7 we elaborate on the commands and their interpretations.

Entry	Interpretation
af	Accounting file path name
br	Communication characteristics: baud rate
lf	Printer error log file
lp	The special print file
ml mx	Lower/upper limit of file size to print A 0 indicates no limits on file size.
pl	Print page length, usually in number of lines
pw	The page width, usually in number of characters
sd	Printer spooling directory
sh	Suppress header page

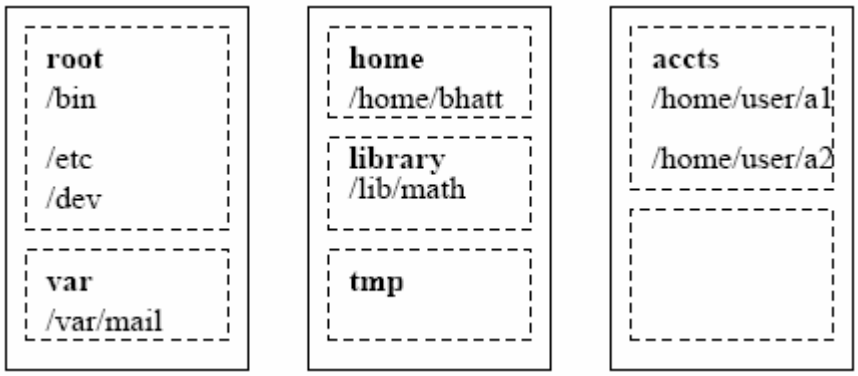
Table 19.7: The printer commands.

To add a printer one may use a *lpadmin* tool. Some of the system administration practices are best learned by assisting experienced system administrators rarely can be taught through a textbook.

19.5.3 Disk space allocation and management

In this section we shall discuss how does a system administrator manage the disk space. We will also like the reader to refer to Section 2.7.1 where we stated that at the time of formatting, partitions of the disk get defined. The partitions may be physical or logical. In case of a physical partition we have the file system resident within one disk drive. In case of logical partition, the file system may extend over several drives. In either of these cases the following issues are at stake:

1. **Disk file system:** In Chapter 2 we indicated that system files are resident in the root file system. Similarly, the user information is maintained in home file system created by the administrator. Usually, a physical disk drive may have one or more file systems resident on it. As an example, consider the mapping shown in Figure 19.1. We notice that there are three physical drives with mapping or root and



The names of file systems are shown in **bold l**etters.

Figure 19.1: Mapping file systems on physical drives.

other file systems. Note that the disk drive with the root file system co-locates the *var* file system on the same drive. Also, the file system home extends over two drives. This is possible by appropriate assignment of the disk partitions to various file systems. Of course, system programmers follow some method in both partitioning and allocating the partitions. Recall that each file system maintains some data about each of the files within it.

System administrators have to reallocate the file systems when new disks become available, or when some disk suffers damage to sectors or tracks which may no longer be available.

2. **Mounting and unmounting:** The file systems keep the files in a directory structure which is essentially a tree. So a new file system can be created by specifying the point of mount in the directory tree. A typical *mount* instruction has the following format.

   mount a-block-special-file point-of-mount

   Corresponding to a *mount* instruction, there is also an instruction to unmount. In Unix it is *umount* with the same format as *mount.*

   In Unix every time we have a new disk added, it is mounted at a suitable point of mount in the directory tree. In that case the mount instruction is used exactly as explained. Of course, a disk is assumed to be formatted.

3. Disk quota: Disk quota can be allocated by reconfiguring the file system usually located at */etc/fstab*. To extend the allocation quota in a file system we first have to modify the corresponding entry in the */etc/fstab* file. The system administration can set hard or soft limits of user quota. If a hard limit has been set, then the user simply cannot exceed the allocated space. However, if a soft limit is set, then the user is cautioned when he approaches the soft limit. Usually, it is expected that the user will resort to purging files no longer in use. Else he may seek additional disk space. Some systems have quota set at the group level. It may also be possible to set quota for individual users. Both these situations require executing an edit quota instruction with user name or group name as the argument. The format of *edquota* instruction is shown below.

   edquota user-name

4. Integrity of file systems: Due to the dynamics of temporary allocations and moving files around, the integrity of a file system may get compromised. The following are some of the ways the integrity is lost:

   - Lost files. This may happen because a user ahs opened the same file from multiple windows and edited them.
   - A block may be marked free but may be in use.
   - A block may be marked in use but may be free.
   - The link counts may not be correct.
   - The data in the file system table and actual files may be different.

The integrity of the file system is checked out by using a *fsck* instruction. The argument to the command is the file system which we need to check as shown below.

fsck file-system-to-be-checked

On rebooting the system these checks are mandatory and routinely performed. Consequently, the consistency of the file system is immediately restored on rebooting.

5. Access control: As explained earlier in this chapter, when an account is opened, a user is allocated a group. The group determines the access. It is also possible to offer an initial set-up that will allow access to special (licensed) software like matlab suite of software.
6. Periodic back-up: Every good administrator follows a regular back-up procedure so that in case of a severe breakdown, at least a stable previous state can be achieved.

19.6 After-Word

In this moduler we have listed many tasks which system administrators are required to perform. However, as we remarked earlier, the best lessons in system administration are learned under the tutelage of a very experienced system administrator. There is no substitute to the “hands-on" learning.