Chapter 3
General Purpose Utilities

Vivekanand College of BCA

How to Login (Login)

When you first connect to a UNIX system, you usually see a prompt such as the following:

```
login:
```

To login the UNIX system:

1. Have your userid (user identification) and password ready. Contact your system administrator if you don't have these yet.
2. Type your userid at the login prompt, then press ENTER. Your userid is case-sensitive, so be sure you type it exactly as your system administrator instructed.
3. Type your password at the password prompt, then press ENTER. Your password is also case-sensitive.
4. If you provided correct userid and password then you would be allowed to enter into the system. Read the information and messages that come up on the screen something as below.

```
login : amrood
amrood's password:
Last login: Sun Jun 14 09:32:32 2009 from 62.61.164.73
$
```

You would be provided with a command prompt (sometime called $ prompt) where you would type your all the commands.

Change Password(passwd):

All Unix systems require passwords to help ensure that your files and data remain your own and that the system itself is secure from hackers and crackers. Here are the steps to change your password:

1. To start, type **passwd** at command prompt as shown below.
2. Enter your old password the one you're currently using.

Vivekanand College of BCA

3. Type in your new password. Always keep your password complex enough so that no body can guess it. But make sure, you remember it.

4. You would need to verify the password by typing it again.

```
$ passwd
Changing password for amrood
(current) Unix password:******


New UNIX password:*******
Retype new UNIX password:*******
passwd: all authentication tokens updated successfully
$
```

date and time (date)

the date command displays the current date and time, including the abbreviated day name, abbreviated month name, day of the month, the time separated by colons, the time zone name, and the year.

Example: display the current date and time

```
$ date
Fri Jul 27 14:12:06 EDT 2007
```

The security feature of UNIX is the command doesn't prompt you to change either date or time. This facility is available only to administrator.

Format specifier in date command

The date command can also used with suitable format specifiers as arguments. Each format is preceded by the + symbol, followed by the %operator, and a single character describing the format.

Example: display the current month only

```
$ date +%m
08
```

Example : display the month name

```
$ date +%h
Aug
```

Example: combine month with it's name in one command

```
$ date +"%h %m"
Aug 08
```

Following are the various format specifiered used with date command.

Format spcefiers	Meaning
d	The day of the month (1 to 31)
y	The last two digits of the year
H, M and S	The hour, minute and second respectively
D	The date in the format mm/dd/yy
T	The time in the format hh:mm:ss
Y	Year in 4 digit

When you use multiple format spcifiers, you must enclose them within quotes (single or double), and use a single + symbol before it.

Clear the screen (tput clear)

UNIX system offers tput command (with clear option) to clear the screen.

```
$ tput clear
$
```

The screen clears and the prompt and cursor are positioned at the top-left corner.

Vivekanand College of BCA

View calendar (cal)

Cal command is used to see calendar of any specific month or a complete year.

The following are the option which we can use with cal command.
OPTIONS:

-1	Displays single month as output.
-3	Displays prev/current/next month output.
-s	Displays sunday as the first day of the week.
-m	Displays Monday as the first day of the week.
-j	Displays Julian dates (days one-based, numbered from January 1).
-y	Displays a calendar for the current year.

Example 1:

To check the calendar for the current month of year.

```
$ cal
     June 2009
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

$
```

Example 2:

To see the calendar for specific month, for example month of Sepember 2008, provide the month number and year as the two arguments to cal command:

```
$ cal 9 2008
   September 2008
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Example 3:

```
$ cal -3 5 2008
        April 2008              May 2008              June 2008
      Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3  4  5                 1  2  3   1  2  3  4  5  6  7
       6  7  8  9 10 11 12   4  5  6  7  8  9 10   8  9 10 11 12 13 14
      13 14 15 16 17 18 19  11 12 13 14 15 16 17  15 16 17 18 19 20 21
      20 21 22 23 24 25 26  18 19 20 21 22 23 24  22 23 24 25 26 27 28
      27 28 29 30           25 26 27 28 29 30 31  29 30
```

Here the cal command displays the calendar of April, May and June month of year 2008.

Displaying a Message (echo)

echo command is used specially for two purpose.

- To display a message (echo hello)
- To evaluate shell variables (echo $a)

Escape sequences in echo command

An escape sequences is generally a two character-string beginning with a \ (backslash).

- \c it places the cursor and prompt in the same line that display the output.

Example:

```
$ echo -e "Enter filename : \c"

Enter filename : $ _
```

- \t A tab which pushes text to the right by eight character positions.
- \n A newline which creates the effect of pressing (*Enter*).

All escape sequences are not two-character strings. ASCII characters can also be represented by their octal values. echo interprets number as octal when it is preceded by \0.

Vivekanand College of BCA

```
$ echo '\07'

…… beep heard …….
```

Options

-n	do not output the trailing newline.
-e	enable interpretation of backslash escapes.
-E	disable interpretation of backslash escapes (default).
--help	display a help message and exit.
--version	output version information and exit.

If –e is in effect, the following sequences are recognized.

Table : Escape sequences used by echo and printf

****	Backslash
\a	alert (BELL)
\b	Backspace
\c	Cursor in same line.
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

\0n	ASCII character represented by the octal value n, where n can't exceed 0377 (decimal value 255)

Display Message, Alternative of Echo (printf)

The printf command is alternative to echo command, used to display message and evaluate variables. It is an external command. But in Bash shell the printf is built-in.

```
$ printf "please enter the correct value\n"
please enter the correct value
$ _
```

printf also accept all escape sequence used by echo. But unlike echo, it doesn't automatically insert a newline unless the \n is used explicitly.

printf also uses formatted strings in the same way the C language used.

Example 1

```
$ printf "My current shell is %s\n" $SHELL
My current shell is /user/bin/bash
```

The %s format string acts as a placeholder for the value of $SHELL, and printf replaces %s with the value of $SHELL.

Format string	Meaning
%s	String
%30s	As above but printed in a space 30 characters wide
%d	Decimal integer
%6d	As above but printed in a space 6 characters wide
%o	Octal integer

Vivekanand College of BCA

%x	Hexadecimal integer
%f	Floating point number

Example 2

```
$ printf "The value of 255 is %o in octal and %x in hexadecimal\n" 255 255
The value of 255 is 377 in octal and ff in hexadecimal
```

The Calculator (bc)

bc command is used to perform arithmetic calculation. If you invoke bc without arguments, the cursor keeps on blinking. It expect input from keyboard when used without an argument.

Example 1:

```
$ bc
12 + 5
17                  Value displayed after computation
[ Ctrl – d ]        The eof character
```

bc shows the output of the computation in the next line.

We can write multiple calculation in the same line, using the ; as delimiter. The output of each computation is, however, shown in a separate line:

```
$ 3 * 4 ; 2 ^ 3                    ^ indicates "to the power of"
12
8
```


Vivekanand College of BCA

bc performs only integer computation and truncates the decimal portion that it sees.

Example : divide two numbers

$ bc	
9 / 5	
1	Decimal portion truncated

To enable floating point computation, you have to set scale the the number of digits of precision before you key in the expression:

$ bc	
scale =2	Truncate to 2 decimal places
17/7	
2.42	rounded off to two digits The
	Actual is 2.42857

bc command is also useful to converting number from one base to another. For example, you can convert binary number to decimal. Set ibase (input base) to 2 before you provide the number:

ibase=2	
11001010	
202	output in decimal – base 10

It is also possible to convert octal to binary

14	
1110	Binary of 14

Vivekanand College of BCA

This way, you can convert from one base to the other. It is also possible for performing scientific calculations.

Recording your Session (Script)

The **script** command makes a typescript of the terminal session.

script makes a typescript of everything printed on your terminal. It is useful for users who need a hardcopy record of an interactive session as proof of work done, as the typescript file can be printed out later with **lpr**. If the *file* argument is given, **script** saves all dialogue in *file*. If no file name is given, the typescript is saved in a file named **typescript**.

-a, **--append**	Append the output to *file* or **typescript**, retaining the prior contents.

The script ends when the forked shell exits (a **control-D** to exit the Bourne shell (sh), and **exit**, **logout** or **control-d** (if **ignoreeof** is not set) for the C-shell, csh).

Certain interactive commands, such as **vi**, create garbage in the typescript file. **script**works best with commands that do not manipulate the screen; the results are meant to emulate a hardcopy terminal.

Example:

```
$ script myfile.txt
```

Logs all results to file **myfile.txt**. This will open a subshell and records all information through this session. The script ends when the forked shell exits (e.g., when the user types **exit**) or when **CTRL-D** is typed.

Your Machine Characteristics (uname)

The uname command displays certain features of the operating system running on your machine.

By default, it simply displays the name of the operating system:

```
$ uname
Linux
```

The Current Release (-r)

-r option is used to display version of your operating system.

The Machine Name (-n)

If your machine is connected to a network, it must have a name called hostname. If your network is connected to the Internet, then this hostname is a component of your machine's domain name (for example merchury.heavens.com).

The –n option tells you the hostname.

```
$ uname –n
Mercury
```

The same output would be obtain with the hostname command.

Knowing your terminal (tty)

tty (teletype) command is used to know the filename of the terminal you are using.

```
$ tty
/dev/pts/10
```

The terminal filename is 10 resident in the pts directory. This directory in turn is under the /dev directory.

Displaying and Setting Terminal Characteristics(stty)

The stty command is used to set the terminal characteristics.

-a option

The –a (all) option displays the current settings.

```
$ stty –a
```

Vivekanand College of BCA

Whether backspacing should erase character (echoe)

If the echoe is set then backspace removes the character from display.

```
$ stty echoe
```

If you want that backspace doesn't removes characters from the display, just reverse the setting. Here you need to prefix a "-" to the echoe keyword.

```
stty –echoe
```

now the backspace doesn't remove character from display.

Entering password through a Shell Script (echo)

When you enter password on the screen in UNIX, it must not be displayed on the screen. It is done by echo option, if you turned it off the keyboard entry is not echoed.

By default it is turned on. It can turned off by the following.

```
$ stty –echo
```

With this setting, keyboard is not echoed.

Changing the Interrupt Key (intr)

stty also sets the function for some of the keys. For example if you like to use [Ctrl-c] as the interrupt key instead of [Delete] then use following command

```
$ stty intr \^c
```

This is the way stty indicates to the system that the interrupt character is [Ctrl –c].

Changing the End-of-File (eof)

When using mailx, you used [Ctrl-d] to terminate input. This eof character also changeable. We can set [Ctrl-a] instead of [Ctrl-d] as the eof character.

```
$ stty eof \^a
```

When Everything Else Fails (sane)

stty also provides another argument to set the terminal characteristics to values that will work on most terminals.

Vivekanand College of BCA

$ stty sane	Restore sanity to the terminal

Who Are You? (whoami)

While you're logged in to the system, you might be willing to know : **Who am I**?

The easiest way to find out "who you are" is to enter the **whoami** command:

```
$ whoami
amrood
$
```

Try it on your system. This command lists the account name associated with the current login. You can try **who am i** command as well to get information about yourself.

Who is Logged In?(users,who,w)

Sometime you might be interested to know who is logged in to the computer at the same time. There are three commands are available that gives you user's information, based on how much you'd like to learn about the other users: **users, who,** and **w**.

users command print the user names of users currently logged in to the current host

```
$ users
amrood bablu qadir
```

who command displays an informative listing of these users.

```
$ who
Amrood    ttyp0    Oct 8 14:10    (limbo)
bablu     ttyp2    Oct 4 09:08    (calliope)
qadir     ttyp4    Oct 8 12:09    (dent)
$
```

Vivekanand College of BCA

-H option is used for heading.

```
$ who –H
NAME        LINE        TIME              COMMENTS
Amrood      ttyp0       Oct 8 14:10       (limbo)
bablu       ttyp2       Oct 4 09:08       (calliope)
qadir       ttyp4       Oct 8 12:09       (dent)
$
```

-Hu option -H option, if used with –u option gives detailed list.

```
$ who –Hu
NAME        LINE        TIME              IDLE   PID      COMMENTS
Amrood      ttyp0       Oct 8 14:10       0:48   11040    (limbo)
bablu       ttyp2       Oct 4 09:08       0:33   11200    (calliope)
qadir       ttyp4       Oct 8 12:09       .      13678    (dent)
```

-a option, gives you all information regarding logged in users

```
$ who –aH
NAME        LINE    TIME          IDLE    PID        COMMENTS      EXIT
Amrood      ttyp0   Oct 8 14:10   0:48    11040      (limbo)       term=0 exit=0
bablu       ttyp2   Oct 4 09:08   0:33    11200      (calliope)
qadir       ttyp4   Oct 8 12:09   .       13678      (dent)        term=0 exit=0
$
```

w command

The **w** command is a quick way to see who is logged on and what they are doing.

w displays information about the users currently on the machine, and their processes. The header shows, in this order, the current time, how long the system has been running, how

Vivekanand College of BCA

many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

The following entries are displayed for each user: login name, the tty name, the remote host, login time, idle time, JCPU, PCPU, and the command line of their current process.

The JCPU time is the time used by all processes attached to the tty. It does not include past background jobs, but does include currently running background jobs.

The PCPU time is the time used by the current process, named in the "what" field.

COMMAND-LINE OPTIONS

-h

Don't print the header.

-u

Ignores the username while figuring out the current process and cpu times. To demonstrate this, do a "su" and do a "w" and a "w -u".

-s

Use the short format. Don't print the login time, JCPU or PCPU times.

-f

Toggle printing the **from** (remote hostname) field. The default if for the **from** field to be printed, although your system administrator or distribution maintainer may have compiled a version in which the **from** field is not shown by default.

-V

Display version information.

user

Show information about the specified user only.

Try **w** command on your system to check the output. This would list down few more information associated with the users logged in the system.

Logging Out (logout):

When you finish your session, you need to log out of the system to ensure that nobody else accesses your files while masquerading as you.

To log out:

```
$logout
```

1. Just type **logout** command at command prompt, and the system will clean up everything and break the connection.

The Universal Mailer (mailx)

Sending mail

In the sending mode, mailx turns interactive, prompting for the subject first, before entering the message body.

```
$ mailx tom
Subject: New project
The new project will start functioning from next month.
Get ready for the next project. – Jerry
[Ctr – d]
EOT                                    It indicates EOT
```

The sending message is simple. The sending message doesn't directly appear on tom's terminal but lands in his mailbox, which is usually /var/mail/tom.

Sending Mail Noninteractively

We can use a shell feature called redirection to take the message body from a file and the –s option to specify the subject:

```
$ mailx –s "New Project" tom < message.txt
```

Vivekanand College of BCA

We can send copy to multiple recipients(enclosed in quotes) using –c option.

```
$ mailx –s "New Project" –c "nayan, jay" tom < message.txt
```

This command sends a message to tom with copies to nayan and jay. If this command is placed in a shell script, mail will be sent without user intervention.

Receiving Mail

All incoming mail is appended to the mailbox. This is a text file named after the user-id of the receipient. Unix systems maintain the mailbox in a directory which is usually /var/mail (/var/spool/mail in Linux). Tom's mail is appended to /var/mail/tom. By default, mailx reads this for viewing received mail.

The shell on tom's machine regularly checks his mailbox to determine the receipt of new mail. If tom is currently running a program, the shell waits for program execution to finish before flashing the program message.

```
You have new mail in /var/mail/tom
```

When tom logs in, he may also see this message.To see the mail sent by jerry, tom invoke the mailx command in the receiving mode. The system first displays the headers and some credentials of all incoming mails that's still held in the mailbox

```
$ mailx

Mailx version 5.0 Wed Jan 5 16:00:40 PST 2000 Type ? for help.

"/var/mail/Charlie": 4 messages 2 new 4 unread

U 1 jerry@jack.hill.com        Fri Apr 3  16:38  19/567        "sweat
dreams"

U 2 MAILER-DAEMON@jack.in  Sat Apr 4 16:33  69/2350  "warning: Coulld not
see"
```

Vivekanand College of BCA

```
N 3 MAILER-DAEMON@jack.in   Thu Apr 9 08:31   63/2066   "warning: Coulld not see"

N 4 jerry@jack.hill.com     Thu Apr 9 08:31   19/567    "New Project"

?
```

To view the message of jerry, tom has to simply type message number.

Message 4:

```
From jerry@jack.hill.com          Thu Apr 9 08:31 2003

Date:  Thu Apr 9 08:31 2003

From "jerry Duck" <jerry@jack.hill.com"

To: tom@saturn.heavens.com

The new project will start functioning from next month.

Get ready for the next project. – Jerry

? q

Saved 1 message in /users1/home/staff/tom/mbox
```

After a message has been seen by the receipient, it moves from the mailbox to the mbox, the secondary storage. The name of this file is generally mbox, to be found in the user's home directory.

Vivekanand College of BCA

Listing Directories and Files(ls):

All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

You can use **ls** command to list out all the files or directories available in a directory. Following is the example of using **ls** command with **-l** option.

```
$ ls -l
total 19621
drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml
-rw-rw-r-- 1 amrood amrood 5341 Dec 25 08:38 uml.jpg
drwxr-xr-x 2 amrood amrood 4096 Feb 15 2006 univ
drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia
-rw-r--r-- 1 root root 276480 Dec 9 2007 urlspedia.tar
drwxr-xr-x 8 root root 4096 Nov 25 2007 usr
-rwxr-xr-x 1 root root 3192 Nov 25 2007 webthumb.php
-rw-rw-r-- 1 amrood amrood 20480 Nov 25 2007 webthumb.tar
-rw-rw-r-- 1 amrood amrood 5654 Aug 9 2007 yourfile.mid
-rw-rw-r-- 1 amrood amrood 166255 Aug 9 2007 yourfile.swf
$
```

Here enteries starting with **d.....** represent directories. For example uml, univ and urlspedia are directories and rest of the enteries are files.

Listing Files:

To list the files and directories stored in the current directory. Use the following command:

```
$ls
```

Here is the sample output of the above command:

```
$ls
bin hosts lib res.03
ch07 hw1 pub test_results
ch07.bak hw2 res.01 users
docs hw3 res.02 work
```

Vivekanand College of BCA

The command **ls** supports the **-1** option which would help you to get more information about the listed files:

```
$ls -l
total 1962188
drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml
-rw-rw-r-- 1 amrood amrood 5341 Dec 25 08:38 uml.jpg

drwxr-xr-x 2 amrood amrood 4096 Feb 15 2006 univ
drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia
-rw-r--r-- 1 root root 276480 Dec 9 2007 urlspedia.tar
drwxr-xr-x 8 root root 4096 Nov 25 2007 usr
drwxr-xr-x 2 200 300 4096 Nov 25 2007 webthumb-1.01
-rwxr-xr-x 1 root root 3192 Nov 25 2007 webthumb.php
-rw-rw-r-- 1 amrood amrood 20480 Nov 25 2007 webthumb.tar
-rw-rw-r-- 1 amrood amrood 5654 Aug 9 2007 yourfile.mid
-rw-rw-r-- 1 amrood amrood 166255 Aug 9 2007 yourfile.swf
drwxr-xr-x 11 amrood amrood 4096 May 29 2007 zlib-1.2.3
$
```

Here is the information about all the listed columns:

1. First Column: represents file type and permission given on the file. Below is the description of all type of files.

2. Second Column: represents the number of memory blocks taken by the file or directory.

3. Third Column: represents owner of the file. This is the Unix user who created this file.

4. Fourth Column: represents group of the owner. Every Unix user would have an associated group.

5. Fifth Column: represents file size in bytes.

Vivekanand College of BCA

6. Sixth Column: represents date and time when this file was created or modified last time.

7. Seventh Column: represents file or directory name.

In the ls -l listing example, every file line began with a d, -, or l. These characters indicate the type of file that's listed.

Prefix	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication.

Meta Characters:

Meta characters have special meaning in Unix. For example ***** and **?** are metacharacters. We use ***** to match 0 or more characters, a question mark **?** matches with single character.

Vivekanand College of BCA

Example:

```
$ls ch*.doc
```

Displays all the files whose name start with ch and ends with .doc:

ch01-1.doc ch010.doc ch02.doc ch03-2.doc

ch04-1.doc ch040.doc ch05.doc ch06-2.doc

ch01-2.doc ch02-1.doc c

Here * works as meta character which matches with any character. If you want to display all the files ending with just .doc then you can use following command:

```
$ls *.doc
```

Hidden Files:

An invisible file is one whose first character is the dot or period character (.). UNIX programs (including the shell) use most of these files to store configuration information.

Some common examples of hidden files include the files:

.profile : the Bourne shell (sh) initialization script

.kshrc : the Korn shell (ksh) initialization script

.cshrc : the C shell (csh) initialization script

.rhosts : the remote shell configuration file

To list invisible files, specify the -a option to ls:

```
$ ls -a
.       .profile docs lib test_results
..      .rhosts hosts pub users
```

Vivekanand College of BCA

```
.emacs bin hw1 res.01 work
.exrc ch07 hw2 res.02
.kshrc ch07.bak hw3 res.03
$
```

Single dot . This represents current directory.

Double dot .. This represents parent directory.

Note: I have put stars (*) just to show you the location where you would need to enter the current and new passwords otherwise at your system, it would not show you any character when you would type.

Creating Files:

You can use **vi** editor to create ordinary files on any Unix system. You simply need to give following command:

```
$ vi filename
```

Above command would open a file with the given filename. You would need to press key **i** to come into edit mode. Once you are in edit mode you can start writing your content in the file as below:

```
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
Once you are done, do the following steps:
```

- Press key **esc** to come out of edit mode.

- Press two keys **Shift + ZZ** together to come out of the file completely.

Now you would have a file created with **filemame** in the current directory.

Vivekanand College of BCA

Editing Files:

You can edit an existing file using **vi** editor. We would cover this in detail in a separate tutorial. But in short, you can open existing file as follows:

```
$ vi filename
```

Once file is opened, you can come in edit mode by pressing key **i** and then you can edit file as you like. If you want to move here and there inside a file then first you need to come out of edit mode by pressing key **esc** and then you can use following keys to move inside a file:

l key to move to the right side.

h key to move to the left side.

k key to move up side in the file.

j key to move down side in the file.

So using above keys you can position your cursor where ever you want to edit. Once you are positioned then you can use **i** key to come in edit mode. Edit the file, once you are done press **esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

Display Content of a File:

You can use **cat** command to see the content of a file. Following is the simple example to see the content of above created file:

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
$
```

You can display line numbers by using **-b** option along with **cat** command as follows:

Vivekanand College of BCA

```
$ cat filename -b
1 This is unix file....I created it for the first time.....
2 I'm going to save this content in this file.
$
```

Counting Words in a File:

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is the simple example to see the information about above created file:

```
$ wc filename
2 19 103 filename
$
```

Here is the detail of all the four columns:

1. First Column: represents total number of lines in the file.
2. Second Column: represents total number of words in the file.
3. Third Column: represents total number of bytes in the file. This is actual size of the file.
4. Fourth Column: represents file name.

You can give multiple files at a time to get the information about those file. Here is simple

Example:

```
$ wc filename1 filename2 filename3
```

Copying Files:

To make a copy of a file use the **cp** command. The basic syntax of the command is:

```
$ cp source_file destination_file
```

Vivekanand College of BCA

Following is the example to create a copy of existing file **filename**.

```
$ cp filename copyfile
$
```

Now you would find one more file **copyfile** in your current directory. This file would be exactly same as original file **filename**.

Renaming Files:

To change the name of a file use the **mv** command. Its basic syntax is:

```
$ mv old_file new_file
```

Following is the example which would rename existing file **filename** to **newfile**:

```
$ mv filename newfile
$
```

The **mv** command would move existing file completely into new file. So in this case you would fine only **newfile** in your current directory.

Deleting Files:

To delete an existing file use the **rm** command. Its basic syntax is:

```
$ rm filename
```

Caution: It may be dangerous to delete a file because it may contain useful information. So be careful while using this command. It is recommended to use **-i** option along with **rm** command.

Example which would completely remove existing file **filename**:

```
$ rm filename
$
```

You can remove multiple files at a tile as follows:

```
$ rm filename1 filename2 filename3
$
```

Vivekanand College of BCA

Standard Unix Streams:

Under normal circumstances every Unix program has three streams (files) opened for it when it starts up:

1. stdin : This is referred to as *standard input* and associated file descriptor is 0. This is also represented as STDIN. Unix program would read default input from STDIN.

2. **stdout :** This is referred to as *standard output* and associated file descriptor is 1. This is also represented as STDOUT. Unix program would write default output at STDOUT

3. **stderr :** This is referred to as *standard error* and associated file descriptor is 2. This is also represented as STDERR. Unix program would write all the error message at STDERR.

A directory is a file whose sole job is to store file names and related information. All files whether ordinary, special, or directory, are contained in directories.

UNIX uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree . The tree has a single root node, the slash character (/), and all other directories are contained below it.

Home Directory:

The directory in which you find yourself when you first login is called your home directory.

You will be doing much of your work in your home directory and subdirectories that you'll be creating to organize your files.

You can go in your home directory anytime using the following command:

```
$cd ~
$
```

Here ~ indicates home directory. If you want to go in any other user's home directory then use the following command:

```
$cd ~username
$
```

Vivekanand College of BCA

To go in your last directory you can use following command:

```
$cd -
$
```

Absolute/Relative Pathnames:

Directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname.

Elements of a pathname are separated by a /. A pathname is absolute if it is described in relation to root, so absolute pathnames always begin with a /.

These are some example of absolute filenames.

```
/etc/passwd
/users/sjones/chem/notes
/dev/rdsk/Os3
```

A pathname can also be relative to your current working directory. Relative pathnames never begin with /. Relative to user amrood' home directory, some pathnames might look like this:

```
chem/notes
personal/res
```

To determine where you are within the filesystem hierarchy at any time, enter the command **pwd** to print the current working directory:

```
$pwd
/user0/home/amrood
$
```

Listing Directories:

To list the files in a directory you can use the following syntax:

```
$ls dirname
```

Vivekanand College of BCA

Following is the example to list all the files contained in /usr/local directory:

```
$ls /usr/local
X11 bin gimp jikes sbin
ace doc include lib share
atalk etc info man ami
```

Creating Directories:

Directories are created by the following command:

```
$mkdir dirname
```

Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command:

```
$mkdir mydir
$
```

Creates the directory mydir in the current directory. Here is another example:

```
$mkdir /tmp/test-dir
$
```

This command creates the directory test-dir in the /tmp directory. The **mkdir** command produces no output if it successfully creates the requested directory.

If you give more than one directory on the command line, mkdir creates each of the directories. For example:

```
$mkdir docs pub
$
```

The above creates the directories docs and pub under the current directory.

Vivekanand College of BCA

Creating Parent Directories:

Sometimes when you want to create a directory, its parent directory or directories might not exist. In this case, mkdir issues an error message as follows:

```
$mkdir /tmp/amrood/test
mkdir: Failed to make directory "/tmp/amrood/test";
No such file or directory
$
```

In such cases, you can specify the **-p** option to the **mkdir** command. It creates all the necessary directories for you. For example:

```
$mkdir -p /tmp/amrood/test
$
```

Above command creates all the required parent directories.

Removing Directories

Directories can be deleted using the **rmdir** command as follows:

```
$rmdir dirname
$
```

Note: To remove a directory make sure it is empty which means there should not be any file or sub-directory inside this directory.

You can create multiple directories at a time as follows:

```
$rmdir dirname1 dirname2 dirname3
$
```

Above command removes the directories dirname1, dirname2, and dirname2 if they are empty. The rmdir command produces no output if it is successful.

Vivekanand College of BCA

Changing Directories:

You can use the **cd** command to do more than change to a home directory: You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as follows:

```
$cd dirname
$
```

Here, dirname is the name of the directory that you want to change to. For example, the command:

```
$cd /usr/local/bin
$
```

Changes to the directory /usr/local/bin. From this directory you can cd to the directory /usr/home/amrood using the following relative path:

```
$cd ../../home/amrood
$
```

Renaming Directories:

The mv (move) command can also be used to rename a directory. The syntax is as follows:

```
$mv olddir newdir
$
```

You can rename a directory **mydir** to **yourdir** as follows:

```
$mv mydir yourdir
$
```

The directories . (dot) and .. (dot dot)

The filename . (dot) represents the current working directory; and the filename .. (dot dot) represent the directory one level above the current working directory, often referred to as the parent directory.

Vivekanand College of BCA

If we enter the command to show a listing of the current working directories files and use the -a option to list all the files and the -l option provides the long listing, this is the result.

```
$ls -la
drwxrwxr-x 4 teacher class 2048 Jul 16 17.56 .
drwxr-xr-x 60 root 1536 Jul 13 14:18 ..
---------- 1 teacher class 4210 May 1 08:27 .profile
-rwxr-xr-x 1 teacher class 1948 May 12 13:42 memo
$
```

File ownership is an important component of UNIX that provides a secure method for storing files. Every file in UNIX has the following attributes:

Owner permissions: The owner's permissions determine what actions the owner of the file can perform on the file.

Group permissions: The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.

Other (world) permissions: The permissions for others indicate what action all other users can perform on the file.

The Permission Indicators

While using **ls -l** command it displays various information related to file permission as follows:

```
$ls -l /home/amrood
-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile
drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir
```

Here first column represents different access mode ie. permission associated with a file or directory.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x):

Vivekanand College of BCA

The first three characters (2-4) represent the permissions for the file's owner. For example -rwxr-x**r--** represents that onwer has read (r), write (w) and execute (x) permission.

The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example -rwxr-x**r--** represents that group has read (r) and execute (x) permission but no write permission.

The last group of three characters (8-10) represents the permissions for everyone else. For example -rwxr-x**r--** represents that other world has read (r) only permission.

File Access Modes

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the **read**, **write**, and **execute** permissions, which are described below:

1. Read:

Grants the capability to read ie. view the contents of the file.

2. Write:

Grants the capability to modify, or remove the content of the file.

3. Execute:

User with execute permissions can run a file as a program.

Directory Access Modes:

Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned:

1. Read:

Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.

Vivekanand College of BCA

2. Write:

Access means that the user can add or delete files to the contents of the directory.

3. Execute:

Executing a directory doesn't really make a lot of sense so think of this as a traverse permission. A user must have execute access to the **bin** directory in order to execute ls or cd command.

Changing Permissions (chmod)

To change file or directory permissions, you use the **chmod** (change mode) command. There are two ways to use chmod: symbolic mode and absolute mode.

Using chmod in Symbolic Mode:

The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.

Chmod operator	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.
=	Sets the designated permission(s).

Here's an example using testfile. Running ls -1 on testfile shows that the file's permissions are as follows:

```
$ls -l testfile
-rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
```

Then each example chmod command from the preceding table is run on testfile, followed by ls -l so you can see the permission changes:

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Vivekanand College of BCA

```
$chmod u-x testfile

$ls -l testfile
-rw-rwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod g=r-x testfile
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Here's how you could combine these commands on a single line:

$chmod o+wx,u-x,g=r-x testfile

```
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Using chmod with Absolute Permissions:

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.

Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1	r-x

Vivekanand College of BCA

	(execute) = 5	
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Chmod 761 file1

It will give user the permission read, write and execute, owner read, write and others execute permission on file file1.

Vivekanand College of BCA

Vivekanand College of BCA