Syntax:

```
$ clear
```

3. banner Command: This command displays it argument like poster, with maximum of 10 characters per line.

```
$ banner HELLO
#    # #####  #       #       #####
#    # #      #       #       #    #
#####  ###    #       #       #    #
#    # #      #       #       #    #
#    # #####  #####   #####   #####
```

4. who Command: Unix is a multi-user environment. Unix maintains an account of all the current users of the system. It is good idea to know the people working on the various terminals so that you can send them messages directly. Who command displays the information of all the users currently logged in the system along with their terminal number and the time of login.

```
$ who
kumar    tty01   jan 30 14:09
tiwari   tty02   jan 30 14:15
sharma   tty03   jan 30 13:20
$_
```

By default, who command produces a three columnar output. There are three users of the system, with their login name in first column. The second column shows the device name of their respective terminals. The third column shows the data and time of logging in.

The –h (header) option prints the column headers.

```
$ who
Name      Line   Time
kumar     tty01  jan 30 14:09
tiwari    tty02  jan 30 14:15
sharma    tty03  jan 30 13:20
$_
```

Option	Function
-b	Indicate the time and date of the last reboot.
-h	print line of column headings
-p	print active processes spawned by init
-q	all login names and number of users logged on
-u	list users logged in.

5. who am i Command: The who command when used with the arguments am i, displays a single line of output only i.e. the login details pertaining to the user who invoked this command.

```
$ who am i
kumar        tty01 jan 30 14:09
$_
```

6. man Command: To access this online manual use the man command.

Syntax:

```
man [section_number] page title...
```

Example,

```
$man sort
```

This command would display the pages of the manual that have all the description for the sort command.

7. stty Command: Setting terminal characteristic:

The terminal is the device by which a user communicates with the system. Different terminals are configured differently, depending on the user's choice. stty command allows the user to set, view and control a wide variety of terminal setting. stty can be invoked with the –a (all) option to display all the current settings:

```
$ stty -a
```

In order to view the already programmed key assignments for a particular terminal enter.

```
$ stty -a
erase = ^H; kill = ^U; werase = ^W;
eof = ^; intr = DEL; swtch = ^@
----------------
----------------
$_
```

Now to change the key assignment for a particular control code

Syntax: `stty name_of_the_code name_of_the_key`

Example: to set erase character to ^E or kill to ^K enter:

```
$ stty erase ^E
$ stty kill ^K
$ stty -a
erase = ^E; kill = ^K; werase = ^W;
-------------
-------------
$_
```

9. date Command: The date command is used to print and set the date.

Syntax:

```
date [-u] [+ format]
date [-u] [MMDDHHmm [YY] -t[[CC]YYMMDDhhmm.[SS]]
```

Example:

```
$ date
Tue Aug 28 15:00:21 EST 2003
$_
```

The command can also be used arguments. Each format is preceded by a '+' symbol, followed by '%' operator and a single character describing the format if you can print only month using the format '+%m'.

```
$ date +%m
8
$_
or month name
$ date +%h
Aug
$_
or combine them in one command
$ date +"%h %m"
Aug 28
$_
```

Formatted output for data command are:

Field Descriptor	Function
%a	Abbreviated weekdays (SUN to SAT)
%A	Full weekday name
%b	Abbreviated month name (JAN to DEC)
%B	Full month name
%c	Current date and time
%C	Century (00 to 99)
%d	Day of month 0 to 31

Field Descriptor	Function
%D	Day as MM/DD/YY
%e	Day of month as decimal number
%h	Abbreviated month
%H	Hour 00 to 23
%I	01 to 12 (12 hour clock)
%j	Day of the year 001 to 366
%m	Month of the year 01 to 12
%M	Minute 00 to 59
%n	New line
%p	Am / pm
%r	Time in AM/PM notation
%T	Time in hh:mm:ss (24 hrs clock)
%y	Last two digit of year
%Y	Year (in four digit)

Setting the system date: User invokes the date command with numeric argument to set the system date. This argument is usually an eight character string of the form MMDDhhmm where MM indicates month expressed in two digits, DD is the day, hh represents the hour in 24 hour format and mm the number of minutes.

```
$ date 11032314
Sat Nov 3 24:14:00   EST 1990
$_
```

10. cal Command: cal command is used to display the calendar of any month or year. Any calendar from the year 1 to 9999 can be displayed with this command.

Syntax:

```
$ cal [[month] year]
```

Generally, cal command prints a calendar for the specified year.

If a month is specified then it displays calendar of that month.

```
$ cal 01 2003
      Jan
$ m  tu  w  th  f  s
   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  31
$_
```

The cal command is accurate and also takes into account the leap year adjustments that took place in the year 1752.

option	Function
-1	Display single month output
-3	Display prev / current / next month output
-s	Display Sunday as first day of week
-m	Display Monday as first day of week
-j	Display julian dates
-y	Display a calendar for current year

Describing terminals using termcap and terminfo files:

- Unix is a multiuser Operating System with different terminals connected to the host. Each terminal uses its own set of codes for instance the code sent by one terminal to clear the screen may differ from the code for another terminal.
- Hence the designers of Unix created a collection of terminal descriptions. These descriptions are kept in a system file named **termcap**. The termcap file contains a description, referred to as an entry for each type of terminal. If new terminal is attached, the description of it is added to the termcap file.
- The **termcap** system has some shortcomings. The entries are all stored in one big file and each entry is limited in size (not longer than 1024 characters) to overcome these problems a new systems, terminfo was devised for AT&T system V version of Unix.
  - The terminfo entries area categorized alphabetically and stored in a series of directories.
  - Each terminal description is stored in its own file so they can be accessed faster.

- Terminfo entries can be longer than 1024 characters
- Terminfo entries are stored in a special compressed format that take up less space and is faster for programs to access.

- In multi-user operating system each terminal attached to host through the port. System manager registers the port by specifying the speed of data transmission and reception and its type. In order to check the key assignment for terminal or to change the key assignment according to requirement the stty command is used.

The Unix system start-up and shut down process involves running a complex booting procedure and shut down routine.

II. Listing Files and Directories Commands:

1. ls Command: The ls command displays a list of files which are available in directory.

Syntax:

```
ls [-options] directory
ls [-adirtXCFR] [file .....]
```

Example:

```
$ ls<enter>
chap1
chap2
chap3
$_
```

Options of ls command:

Option	Function
-a	Displays all files including ., .. and hidden files
-A	Displays all but not .and .
-C	List entries by column
-d	List directory entries instead of contents
-l	Use ling listing format
-r	Sort in reverse order
-R	List subdirectory recursively
-s	Prints size of each file in blocks
-S	Sort by file size
-t	Sort by modification time

Option	Function
-u	Sort by access time
-V	Sort by version
-1	List one file per line
-X	List in column format with row wise sorting
-F	Classifies files, executables, directories
-i	Shows inode number of a file

Example:

```
$ ls -l chap1
-rwxr--r--  1          user1      group1   712       jan 20 10:30
chap1
(File mode) (No. of links) (File owner) (Group name) (Size of file)
(Date & time) (File name)
```

2. pwd Command: Checking your current directory

- You can always find out the directory where you are currently placed by using the pwd (**p**resent **w**orking **d**irectory) command. When user logged into the Unix system, Unix automatically places the user in the home directory.
- The name of home directory is a login name itself. If the user is working in login with username "kumar" his home directory is /usr/kumar or /home/kumar

```
$ pwd
/usr/kumar
$_
```

- The pwd command resembles the cd command of MS-DOS when used without argument. Thus pwd tells you that your current directory is /usr/kumar, you can refer to a sub directory progs, either as /usr/kumar/progs or simply as progs.

3. cd Command: Changing directories

- You can move around in the file system by using the cd (**c**hange **d**irectory) command.
- It changes the current directory to the directory to the directory specified as the argument.

Syntax:

```
cd dirname
cd by itself take us to the Home directory
cd ../.. changes directory to parent of parent directory
cd dir2 changes directory to dir2 in the current directory.
```

Example:

```
$ pwd
/usr/kumar
$ cd progs
$ pwd
/usr/kumar/progs
$_
```

4. **mkdir Command: Making a directory**

- The mkdir command just as in MS-DOS, directories can be created by using the mkdir (**make dir**ectory) command.
- The command is followed by the names of the directories to be created.

Syntax:

```
$mkdir <dirname(s)>
```

Example:

```
$ mkdir chap1
$_
```

unlike MS-DOS, however a number of sub-directories can be created by one mkdir command:

```
$ mkdir lesson lesson/chap1 lesson/chap2
```

This creates three sub directories lesson and two subdirectories.

5. **rmdir Command: Removing a directory**

The rmdir (**rem**ove **dir**ectory) command removes or deletes directories.

Syntax: `rmdir dirname(s)`

Example:

```
$ rmdir lesson
$_
```

- You can not delete a directory unless it is empty. The three directories and the sub directories that were just created with mkdir can be removed by using rmdir with a reversed set of arguments.

```
$ rmdir lesson/chap1 lesson/chap2 lesson
```

6. **touch Command:** Touch command creates new but empty files

```
$ touch sample
$ touch a1 a2 a3
```

7. **rm Command: Deleting files:** Files can be deleted with rm (**rem**ove).

Syntax:

```
rm [-fir] filename .....
```

-i interactive, prompts the user before each file is deleted.

-r recursive, rm recursively deletes the entire contents of the specified directories and the directories themselves.

Example:

```
$ rm chap1 chap2 chap3
$_
```

rm won't normally remove a directory, but it can remove files from one.

```
$ rm progs/chap1 progs/chap2
```

you can remove two chapters from progs directory without having to "cd" to it.

You can easily delete all files in directory by

```
$ rm *
$_
```

MS-DOS users beware! When you delete files in this fashion, the system won't prompt you with the message "Are you sure?" or "all files in directory will be deleted?" before removing the files. The $ prompt will return silently, suggesting that the work has been done. When rm is invoked with options, different things start happing. The –i (interactive) option makes the command prompt the user with each filename and a?, before acting on it:

```
$ rm –i chap1 chap2 chap3
chap1 : ? y
chap2 : ? n
chap3 : ? n
$_
```

if you press 'y' then the file will be deleted.

8. cp Command: Coping a file

- The cp (copy) command copies a file or a group of files. Similar to copy command of MS-DOS, cp creates an exact image of the file on the disk with a different name.
- To copy the file chap1 to unit1 you use,

Syntax:

```
cp file1 file2
cp file(s) directory
```

Example:

```
$ cp chap1 unit1
$_
```

If there is only one file to be copied, then the destination can be either an ordinary or a directory file. For example to copy file chap1 to directory progs.

```
$ cp chap1 progs/units
$_
```

If there are groups of files to be copied, then you specify command line: the syntax requires the destination file to be a directory and copies the files into directory.

```
$ cp chap1 chap2 chap3 progs
```

9. mv Command: Renaming Files

- The mv (**move**) command simply renames a file or a group of files.
- mv does not create a copy of file, it merely renames it.

Syntax:

```
mv source dest Or
mv file1 file2 Or
mv directory directory2
```

Example: To rename the file chap1 to man1,

```
$ mv chap1 man1
$_
$ mv chap1 chap2 chap3 progs
$_
```

moves the files chap1, chap2 and chap3 to the progs directory. mv can be used to rename a directory.

10. cat Command: Displaying and creating files

- cat is used to display the contents of file.

Syntax:

```
cat [file .....]
```

- To do that for the file dept.lst, simply specify the filename as the argument:

```
$ cat dept.lst
01    | computer        | 60
02    | information     | 40
03    | electronics     | 60
$_
```

- cat also accepts more than one filename as arguments.

```
$ cat chap1 chap2
```

- The contents of the second file are shown immediately after the first file without any message or header information.
- Cat is normally used for displaying text files only.

cat is also useful for creating a file. Enter the command cat, followed by the '>' character and the filename (for example abc) and press the <enter> key. Enter text messages and then press <control-D>:

```
$ cat>abc
This is simple text file.
hello
<control-D>
$_
```

cat is also useful for appending the contents at the end of file.

```
$ cat>>abc
```

Text is enter at the end of file

```
<control-D>
$_
$ cat abc
```

This is simple text file.

```
hello
Text is enter at the end of file
$_
```

cat > c:\home\student

11. head Command: Displaying the beginning of a file

- The head command, as the name implies, displays the top of the file.
- By default you can only read first ten lines of a file.

Syntax:

```
head<option><filename>
```

Example:

```
$ head emp.lst
```

You can change the number of lines displayed by specifying a number option as shown below:

```
$ head -20 temp
```

It displays first 20 lines of a file.

12. tail Command: Displaying the end of a file

- The tail command displays the end of the file.
- If no line count is given, tail displays the last ten lines of the file.

Example:

```
$ tail  emp.lst
$_
```

18.01.18. Experiment - 2

Aim - To implement Linux Shell Commands

	SHELL COMMAND	PURPOSE	SYNTAX / EXAMPLE	OUTPUT
1.	who am i	gives the details of the current user	whoami	student
2.	who	gives the details of all the users	who	student tty2 2018-01-17 23:21 (/dev/tty2)
3.	pwd (print working directory)	show the current directory	pwd	/home/student
4.	time	show the time of the system	$time	real 0m0.000s user 0m0.000s sys 0m0.000s
5.	cal (calender)	display the calender	$cal	January 2018 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 31

6.	logname	display the name with which the user logged in	logname	student
7.	uname	display the name of the system	uname	Linux
8.	users	display the names of the users	users	student
9.	tty	display the name of the terminal	tty	/dev/pts/0
10.	history	show the log of commands which are typed in terminal	history	516 logname 517 uname 518 users 519 tty 520 history
11.	ls (list)	list of all the existing files	ls	f1 first.sh f2 fnew f3 Fnew ⋮
12.	ls -l	gives details of all the files and directories	ls -l	drwxrwxr-x. 2 student student 4096 Jan 15 01:46 acbd
13.	ls -a	also shows hidden files	ls -a	file1 .a .file2 abc

14.	man	gives details of particular command (implementation)	man	
15.	info	gives details of command	info	
16.	cd	it is used to go to a particular directory	cd files cd / cd ~ cd - cd ...	cd / cd name cd student
17.	mkdir	to make directory	mkdir doc1 mkdir -p doc1/doc2	
18.	touch	to create multiple empty files	touch file1 file2	
19.	cat	to create a file and to insert data	cat > file1 - Hello [ctrl+d cmd.]	cat file1 - Hello
20.	cat	to display data of a file	cat file1	- Hello - Hi
21.	rmdir	remove directory	rmdir -i doc1	do you really want to delete? (n/y) y

22.	mv	rename an existing file	mv file1 file2	
23.	rm	to delete the file	rm -i file1	do you really want to delete? (n/y) y
24.	wc	details of words into a file	wc file1	1 4 32 file1
25.	cat	to transfer data of one file to another	cat f1 f2 > f3	
26.	cp	copy one file into another	cp file1 file2 cp -r dir1 dir2	cat file2 Hello Hi
27.	head	view contents of files	head file1 head -3 file1	Hello Hel
28.	tail	view contents of files from	tail file1 tail -3 file1	you
29.	tac	displays the contents of a file in reverse order	tac file1	olleH
30.	sort	displays the contents of a file in sorted manner	sort file1	are How you

31.	tr	used to translate the contents of a file in a particular format	cat file1 | tr 'e' 'E'	hEllo how arE you
32.	cut	cuts a specific portion of a file	cut -c3 file1	llo
33.	cat	creation of hidden files	cat >.filen Hello Secret file [ctrl + D]	cat .filen Hello secretfile
34.	bc	to create a calculator mode	bc↵ 2*2 4*1 180/3 quit↵	 4 4 60
35.	ulimit	maximum available space information	ulimit	unlimited
36.	ps	gives information about processor which are running	ps	PID TTY 2283 pts/0 2280 pts/0
37.	ps -f	gives more details	ps -f	
38.	grep	It is used to search keyword	grep -v abc file1 file2	

16

Durga (1)
25/01/18