Contents

COMMANDS

1) cal:- Displays a calendar

Syntax:- cal [options] [month] [year]

Description :-

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

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

Example:-

$cal

or

$cal 02 2016

Feb 2016

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					

2) clear :- It clears the terminal screen.

Syntax :- clear

Description :-

- Clear clears your screen if this is possible, including its scroll back buffer.

- Clear ignores any command-line parameters that may be present.

3) pwd **:-** Displays path from root to current directory

Syntax :- pwd [options]

Example:

$ pwd

/home/kumar/progs

4) cd:- It is used to change the directory.

Syntax :- cd [directory]

Description:-

- Used to go back one directory on the majority of all UNIX shells. It is important that the space be between the cd and directory name or ..

Example:-

$ pwd

/home/kumar

$ cd progs

$ pwd

/home/kumar/progs

$ cd ..

/home/kumar

5) ls:- Lists the contents of a directory

Syntax :- ls [options]

Description :-

-a	Shows you all files, even files that are hidden (these files begin with a dot.)
-A	List all files including the hidden files. However, does not display the working directory (.) or the parent directory (..).
-d	If an argument is a directory it only lists its name not its contents
-l	Shows you huge amounts of information (permissions, owners, size, and when last modified.)
-p	Displays a slash (/) in front of all directories
-r	Reverses the order of how the files are displayed
-R	Includes the contents of subdirectories

Example:-

$ ls – l

-rw-r----- 1 student student 23 Jan 16 15:27 file1.txt

Field Explanation:

- If first character is – then it is normal file
- If it is d then it is directory
- **Field 1 – File Permissions**: Next 9 character specifies the files permission. Each 3 characters refers to the read, write, execute permissions for user, group and world In this example, -rw-r—– indicates read-write permission for user, read permission for group, and no permission for others.
- **Field 2 – Number of links**: Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file.
- **Field 3 – Owner**: Third field specifies owner of the file. In this example, this file is owned by username 'student'.
- **Field 4 – Group**: Fourth field specifies the group of the file. In this example, this file belongs to "student' group.
- **Field 5 – Size**: Fifth field specifies the size of file. In this example, '13' indicates the file size.
- **Field 6 – Last modified date & time**: Sixth field specifies the date and time of the last modification of the file. In this example, 'Jan 16 15:27' specifies the last modification time of the file.
- **Field 7 – File or directory name**: The last field is the name of the file or directory. In this example, the file name is file1.txt

6) exit :- It is used to terminate a program, shell or log you out of a network normally.

Syntax :- exit

7) echo :- It prints the given input string to standard output.

Syntax :- echo string

Example:-

$ echo "hi.. hello unix"

hi.. hello unix

8) who :- who command can list the names of users currently logged in, their terminal, the time they have been logged in, and the name of the host from which they have logged in.

Syntax :- who [options] [file]

Description:-

am i	Print the username of the invoking user, The 'am' and 'i' must be space separated.
-b	Prints time of last system boot.
-d	print dead processes.
-H	Print column headings above the output.
-l	Include idle time as HOURS:MINUTES. An idle time of . indicates activity within the last minute.
-m	Same as who am i.
-q	Prints only the usernames and the user count/total no of users logged in.

Example :-

$ who

dietstaffpts/1 2016-02-20 22:42 (:0.0)

dietstaffpts/2 2016-02-20 09:30 (:0.0)

- Here first column shows user name, second shows name of the terminal the user is working on. Third& fourth column shows date and time of logging, last column shows machine name.

9) who am i:- Print effective userid

Syntax :- who am i

Description: - Print the user name associated with the current effective user id.

Example :-

$ who am i

dietstaffpts/3 2016-02-10 08:52 (:0.0)

- Here first column shows user name, second shows name of the terminal the user is working on. Third & fourth column shows date and time of logging, last column shows machine name.

10) mkdir:- This command is used to create a new directory

Syntax :- mkdir [options] directory
Description :-

-m	Set permission mode (as in chmod)
-p	No error if existing, make parent directories as needed.
-v	Print a message for each created directory
directory	The name of the directory that you wish to create

Example:-

$ mkdir aaa

- The above command will create directory named aaa under the current directory. We can also create number of subdirectories with one mkdir command.

11) rmdir:- It is used to delete/remove a directory and its subdirectories.

Syntax :- rmdir [options..] Directory
Description :-

- It removes only empty directory.

-p	Allow users to remove the directory and its parent directories which become empty.

12) bc:- bc command is used for command line calculator. It is similar to basic calculator. By using which we can do basic mathematical calculations.

Syntax :- bc [options]
Description :-

- bc is a language that supports arbitrary precision numbers with interactive execution of statements.
- bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read.

-q	To avoid bc welcome message

-l	To include math library functionalities

Example:-

$ bc

bc 1.06 Copyright 1991-1994,1997,1998,2000 Free Software Foundation,Inc. This is free software with ABSOLUTELY NO WARRANTY. For details type `warranty'. 2*3 6

- The above command used is for mathematical calculations.

$ bc -l

bc 1.06 Copyright 1991-1994,1997,1998,2000 Free Software Foundation,Inc. This is free software with ABSOLUTELY NO WARRANTY.For details type `warranty'.11+2 13

- The above command displays the sum of '11+2'.

$ bc calc.txt

bc 1.06 Copyright 1991-1994,1997,1998,2000 Free Software Foundation,Inc. This is free software with ABSOLUTELY NO WARRANTY. For details type `warranty'. 13

- 'calc.txt' file have the following code:11+2. Get the input from file and displays the output.

SHELL SCRIPTS

1) Write a shell script to scans the name of the command and executes it.

Program :-

```
echo "enter command name"
read cmd
$cmd
```

Output :-

```
enter command name
cal
February 2016
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
```

2) Write a shell script Which works like calculator and performs below operations Addition , Subtract ,Division ,Multiplication

Program :-

i) using if..elif statement

```
echo " Enter one no."
read n1
echo "Enter second no."
read n2

echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
if [ $ch = "1" ]
then
   sum=`expr $n1 + $n2`
    echo "Sum ="$sum
elif [ $ch = "2" ]
then
      sum=`expr $n1 - $n2`
    echo "Sub = "$sum
elif [ $ch = "3" ]
then
   sum=`expr $n1 \* $n2`
    echo "Mul = "$sum
elif [ $ch = "4" ]
then
   sum=`expr $n1 / $n2`
```

```
    echo "Div = "$sum
fi
```

Output :-

```
Enter one no.
32
Enter second no.
12
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
2
Sub = 20
```

ii) using while loop and switch statement

```
i="y"
while [ $i = "y" ]
do
echo " Enter one no."
read n1
echo "Enter second no."
read n2
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
case $ch in
   1)sum=`expr $n1 + $n2`
    echo "Sum ="$sum;;
    2)sum=`expr $n1 - $n2`
    echo "Sub = "$sum;;
   3)sum=`expr $n1 \* $n2`
```

```
    echo "Mul = "$sum;;
   4)sum=`expr $n1 / $n2`
    echo "Div = "$sum;;
   *)echo "Invalid choice";;
esac
echo "Do u want to continue ? y/n"
read i
if [ $i != "y" ]
then
   exit
fi
done
```

Output :-

```
Enter one no.
32
Enter second no.
22
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
2
Sub = 10
Do u want to continue ? y/n
N
```

3) Write a shell script to print the pyramid structure for the given number.

Program :-

```
echo "enter the number"
read n
printf "\n"
for((i=1;i<=$n;i++))
do
   for((j=1;j<=$i;j++))
```

```
do
   printf "$j"
done
   printf "\n"
done
```

Output :-

```
enter the number
4
1
12
123
1234
```

4) Write a shell script to find the largest among the 3 given numbers.

Program :-

```
clear
echo "Enter first number: "
read a
echo "Enter second number: "
read b
echo "Enter third number: "
read c

if [ $a -ge $b -a $a -ge $c ]
then
        echo "$a is largest integer"
elif [ $b -ge $a -a $b -ge $c ]
then
        echo "$b is largest integer"
elif [ $c -ge $a -a $c -ge $b ]
then
        echo "$c is largest integer"
fi
```

Output :-

Enter first number:
22
Enter second number:
33
Enter third number:
42
44 is largest integer

5) Write a shell script to find factorial of given number n.

Program :-

```
clear
fact=1
echo "Enter number to find factorial : "
read n
a=$n
#if enter value less than 0
if [ $n -le 0 ]
then
echo "invalid number"
exit
fi
#factorial logic
while [ $n -ge  1 ]
do
  fact=`expr $fact \* $n`
  n=`expr $n - 1`
done
echo "Factorial for $a is $fact"
```

Output :-

Enter number to find factorial :
5
Factorial for 5 is 120

INDEX