INDEX

NAME: Hitika Tripathy STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: _____

Commands of UNIX/LINUX-

mkdir -

- It stands for "make Directory".
This command allows users to create or make new directories.
This command also set permissions, create multiple directories (folders) at a time.

Syntax-

mkdir [option] directory_name

Eg.-

mkdir d1 → make single directory.

mkdir -p d1/d2/d3 → make multiple directories.

as → d1
↳ d2
↳ d3

To see all the ~~commands~~ options use man mkdir.

2. Touch-

This command is used to create blank file.
This command also change and modify timestamps of a file.

syntax-

touch filename1 filename2 filename3

Eg.-

touch f1 → create f1 in directory folder.

3. cd -
It stands for "change directory".
Syntax -
cd directory_name
or
cd options
Eg. and options -
cd d1 → go to d1 directory
cd .. → go to parent directory
cd / → go to root directory
cd . → go change to current directory.

4. rmdir -
• It stands for "remove directory".
This command is used to remove ~~empty~~ directories from the filesystem.
Syntax -
rmdir (option) directory
~~Eg. and~~ options -
rm - remove file
rmdir - remove directory
rm -i - remove file interactively
rm - f - remove file forcefully
rm - r - remove file recursively
rm f* - remove all files starting with f.
Eg. -
rmdir -p d1 → remove dir d1 as it is empty
rmdir -- version → to know version of rmdir

5. date -
This command is used to display the date and current time of system.
Syntax -
date (option) ... (+format)
Eg. -
date → display date & time in UTC format.

6. Time -
This command is used to display time of the system.
Syntax -
time (option) ... (command)
Eg. -
time -p → print time in posix format

7. who -
This command is used to print information about users who are currently logged in.
Syntax -
who (option) ... (file | Arg1 Arg2)
Eg. -
who

. who am i -
This command is used to print the user name associated with the current effective user id.
Syntax -
whoami (option) ...
Eg. - whoami

9. ls -
This command is used to list information about the files.
Syntax - ls (option) ... (file) ...
Eg. & options -
ls, ls -a, ls -i, ls -l, ls -r, ls -n, ls -o, ls -s, ls -R.

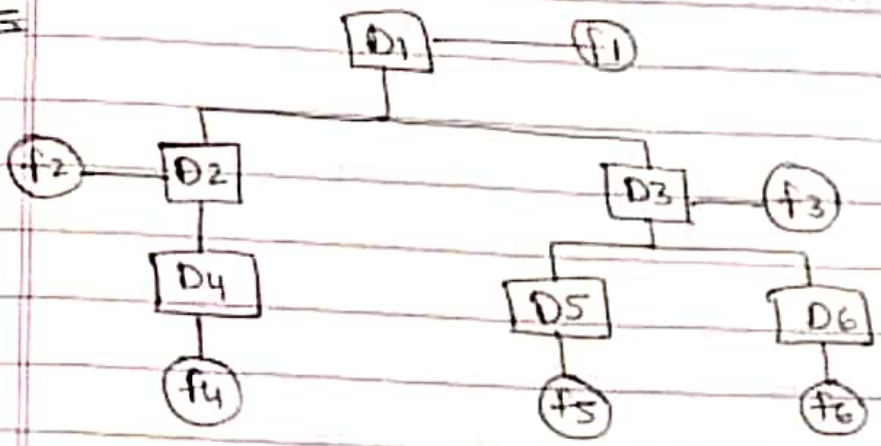
10. banner -
This command represents the given input in fancy way on screen.
Syntax - banner string

11. cal -
This command displays the calender of current year.
Syntax - cal
Eg. - cal 11 2002 → shows november's calender

12. tty -
This command is used to print the file name of the terminal connected to standard output.
Syntax - tty (option)
Eg. - tty --version
tty

13. pwd -
It stands for "Present working Directory".

Syntax- pwd [-lp]
Eg.- pwd



mkdir -p D1/D2/D4 D1/D3/D5 D1/D3/D6
cd D1
touch f1
cd D2
touch f2
cd D4
touch f4
cd
cd D1
cd D3
touch f3
cd D5
touch f5
cd D6
touch f6

14. Ls
This command is used to view the hidden files.
Syntax- ls (option) ... (files) ...

15. cat -
This stands for "Concatenation".
Options-
cat > f1 → this command is used to create file & also save content in file.
cat >> f1 → This command does not erase the data already stored in file & adds new data to existing data.
cat f1 → display the content of file.
cat f1 f2 → concat data of two files and display on screen.
cat f1 f2 > f3 → concat data & store in f3. if any data is there in f3 it will be erased.
cat f1 f2 >> f3 → concat data & store in f3 if data is already present, it will not be deleted.
cat f1 > f2 → ~~concat~~ copy data of f1 to f2 and will overwrite data in f2.
cat f1 >> f2 → copy data of f1 to f2 without deleting old data in f2.

16. cp -
Copy command.
cp f1 f2 → copy data of f1 to f2.

17. move -
move data from one file to another.
move f1 f2 → move data from f1 to f2

18. sort-

sort file and display contents on screen. (in ascending order by default)

options and eqs-

sort f1 → sort & display on screen. original data is not sorted.

sort f1 -o f2 → copy content of f1 & sorted data is stored in f2.

sort -r f1 → sort data in descending order

sort -u f1 → it will ~~show~~ do not show duplicate data.

sort -m f1 f2 → sort & merge f1 & f2.

sort -rm f1f2 → reverse merging

sort -M → sort according to months

sort -n f1 → sort numeric values.

19. grep-

This command is used to search any string or substring in a file.

Syntax-

grep (options) filename.

Options & eg.-

grep man f1 → will search word 'man' in file f1. It will show all ~~words~~ strings which contains 'man'.

grep -i → it is case insensitive.

grep -v man f1 → will display all the string which does not contain 'man'.

grep -n man f1 → display content of string containing 'man' with line number.

grep ^man f1 → display all the strings which starts with man.

grep man$ f1 → display all strings which ends with man.

20. head-

Syntax & example-

head f1 → display initial 10 lines of f1.

head -20 f1 → display first 20 lines of f1.

21. tail-

Syntax & eg.-

tail f1 → display last 10 lines of f1.

tail -20 f1 → display last 20 lines of f1.

22. pg-

display 1 page at a time, ie. full screen.

Syntax & eg.- ~~pag~~ pg f1

23. more-

display 1 page and than line by line.

Syntax & eg.- more f1.

24. less-

same as more

Syntax & eg.- less f1

25. wc -
stands for word count.
It tells us the number of count.
Options & eg. -
wc -l → print no. of lines
wc -w → print no. of words.
wc - c → print no. of characters.
wc fi → print no. of ~~files~~ lines. words & characters

26. expr -
it is expression command.
Eg. & Syntax -
expr 2 + 5
expr 2 / 2
expr 2 * 2 → for multiplication.
expr 2 + 3 - \(4 + 5 \)

27. bc -
It stands for binary calculator.
In bc command -
obase means 10
ibase means 2

28. factor -
It displays prime factor of any number.
Syntax & eg. -
factor 12 → display 1, 2, 3, 4, 6, 12.

29. ln -
It means link.
It creates multiple links of any file
Syntax & eg. -
ln f1 f2 → f1 is existing file & f2 is non-existing. ln creates aliases

30. type -
Internal or external command.

31. chmod -
It is used to change permission of owner, group & user.
Syntax & eg. -
chmod g+w f1
chmod o-r f1.

32. umask -
It will hinder specified permission value to change default permission.

Process Handling Commands -

33. Kill -
This command is used when we want to commit any process before its completion.
Syntax & eg. -
Kill process-name;
Kill column-A;

34. nice-
This command is used to change the priorities of values.
syntax & eg.-
nice pi
nice -5 pi.

35. nohup-
It stands for no hang up.
This command is used to enter in hypernate state.

36. pi t.
it means that pi process will become background process.

37. tar-
this command is used to compress a file.
Syntax-
tar (options) file name.
Example-
tar f1.

3. untar-
This command is used to uncompress a file.

39. sleep-
This command is used to pause the execution of the shell command (current) for a given time.

40. read-
r - it gives permission to read only.

41. write-
w- it gives read-write permission.

42. ps-
This command gives the list of currently executing (running) processes and some other information too.
Syntax & ~~process~~ - example-
ps -f → gives you full string.
pid → process id
ppid → parent process id.

43. vi -
This command opens vi editor.

44. sh-
This command run shell program.

4 Piping -

45. $ ls -l | more → ~~Data~~
Data flow from left to right.

46. $ sort file.txt |uniq, -
first sort file.txt, then find non-duplicate values.

47. echo -
This command display line of text/ string passed as argument.

Eg. -
$ echo "Hi, Welcome"
Hi, Welcome
$ echo -e "How \v are \v you?"
How
are
you?

$ echo * → same as ls.

1. Menu driven program to perform various arithmetic operations (using if statement)

```
echo "enter 2 numbers"
read a
read b
echo "enter operator"
read opt
if [ $opt = "+" ]
then
   v = $((a+b))
elif [ $opt = "-" ]
then
   v = $((a-b))
elif [ $opt = "*" ]
then
   v = $((a*b))
elif [ $opt = "/" ]
 then
   v = $((a/b))
elif [ $opt = "%" ]
 then
   v = $((a%b))
fi
echo $v
```

op-

```
enter 2 numbers
3
4
enter operator +
7
```

2. Menu driven program to perform various arithmetic operations (using case statement)

```
echo "enter two numbers"
read a
read b
echo " enter choice"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch
case $ch in
    1). res = `echo $a+$b | bc`
        ;;
    2). res = `echo $a-$b | bc`
        ;;
    3). res = `echo $a \* $b | bc`
        ;;
    4). res = `echo "scale=2; $a / $b" | bc`
        ;;
esac
echo "Result : " $ res
```

Output -

Enter two numbers	3. Multiplication
2	4. Division.
3	2
Enter choice	Result : -1
1. Addition	
2. Subtraction	

3. Find the maximum and minimum of three given numbers.

```
echo "enter three numbers".
read a
read b
read c
if [ $a -gt $b ] && [ $a -gt $c ]
then
    echo " a is maximum".
elif [ $b -gt $c ] && [ $b -gt $a ]
then
    echo " b is maximum".
else
    echo " c is maximum".
fi
```

Output -

```
enter three numbers
9
6
8
a is maximum.
```

4. Check whether the number is even or odd.

```
echo "enter number"
read n.
r = $ (( $n %.2 ))
if [ $r -eq 0 ]
then
    echo $n " is even number".
else
    echo $n " is odd number".
fi
```

Output -

```
enter number
7
7 is odd number.
```

5. Check whether the number is positive / negative -

```
echo "enter a number"
read n
if [ $n -gt 0 ]
then
    echo $n " is positive".
elif [ $n -lt 0 ]
then
    echo $n " is negative".
else
    echo $n " is zero".
fi
```

Output -

```
enter a number
-8
-8 is negative.
```

6. Generate table of a given number.

```
echo "enter a number"
read n
i=0
while [ $i -lt 10]
do
    echo "$n x $i = `expr $n \* $i`"
    i=`expr $i + 1`.
done
```

Output- enter a number 5

5 x 0 = 0

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

7. Find factorial of a given number.

```
echo "enter a number"
read num
fact=1
while [ $num -gt 1 ]
do
    fact=$((fact * num))
    num=$((num-1))
done
echo "factorial = " $fact
```

Output -

```
enter a number
5
factorial = 120
```

8. Check number is prime or not.

```
echo " enter a number"
read n.
i=2
flag=0
while test $i -lc `expr $n/2`
do
   if test `expr $n %. $i` -eq 0
then
      flag=1
fi
i=`expr $i+1`
done
if test $flag -eq 1
then
   echo " The number is not prime".
else
   echo " The number is prime".
fi.
```

Output -

enter a number.

2

The number is prime.

9. Check whether the no. is palindrome or not.

```
echo "enter number"
read num
rem=0
rev=0
n= $num
while [ $num -gt 0]
do
    rem = $((num %10))
    num = $(( num /10 ))
    rev = $(( rev * 10 + rem ))
done
if [ $n -eq $rev]
then
    echo " number is palindrome"
else
    echo " number is not palindrome"
fi
```

Output -

```
enter number
131
number is palindrome.
```

10. Check whether the number is armstrong or not.

```
echo "enter a number"
read a
x = $a
sum = 0
r = 0
n = 0
while [ $x -gt 0]
do
    r = `expr $x %. 10`
    n = `expr $r \* $r \* $r`
   sum= `expr $sum + $n`
     x = `expr $x / 10`.
done
if [ $sum -eq $c]
then
    echo " Number is armstrong".
else
    echo "Number is not armstrong".
fi
```

Output-

Enter a number.

153

Number is armstrong.

11. Shell script to generate fibonacci series.

```
echo " number of terms to be generated"
read n
 x=0
 y=1
 i=2
echo "Fibonacci series".
echo $x
echo $y
while [$i -lt $n]
do
    i='expr $i+1'
    z='expr $x + $y'
    echo $z
    x=$y
    y=$z
done.
```

Output-

number of terms to be generated.

5

fibonacci series

0

1

1

2

3

12. Shell program for sorting a se of number. The set of numbers are to be entered through file.

```
file = " "
echo -n. " Enter filename"
read file
if [ 1 -f $file ]
then
    echo " $file not a file 1"
    exit 1
fi
sort -n $file
sort -n $file
```

Output -

```
Enter filename
f1
1
12
32
45
100
```

13. Shell script for generation and summation of natural number.

```
echo "Enter size"
read N
i=1
sum=0.
while [ $i -le $N]
do
    num = $((sum + i))
    i = $((i+1))
done
echo "sum = " $sum
```

Output-

Enter size

5

sum = 15

14. Shell program to generate and sum of all prime numbers between any 2 given numbers.

```
echo " enter two numbers".
read n
read m.
sum = 0
for (( i=n ; i<=m ; i++))
do
  for (j=2; j<=$(( i/2)) ;j++))
  do
    if [$(( i%j)) -ne 0]
    then
        echo $i
        sum = $((sum + i))
    fi
  done
done
echo "sum = " $sum.
```

Output -

```
enter 2 numbers
1
7
1
2
3
5
7
sum = 18.
```

15. Pattern

```
*
* *
* * *
* * * *
```

```
i=1
while [ $i -le 4 ]
do
  j=1
  while [ $j -le $i ]
  do
    echo "* \c"
    j=$((  $j +1))
  done
  echo "\n"
  i=$((  $i +1))
done
fi
```

Output -

```
*
* *
* * *
* * * *
```

16.
```
1
1 2
1 2 3
1 2 3 4
i=1
while [ $i -le 4 ]
do
  j=1
  while [ $j -le $i ]
   do
   echo $j "\c"
    j= $ (( $j + 1 ))
    done
     echo "\n"
    i = $ (( $i + 1 ))
  done
fi
```

Output -

```
1
1 2
1 2 3
1 2 3 4
```

17.

```
*
* *
* * *
* * * *
* * *
* *
*
```

```
i=1
while [$1 -le 4]
do
  j=1
  while [$j -le $i]
  do
    echo "*\c"
    j=$((($j+1))
    done
    echo "\n".
  i=$(($i+1))
 done
i=3
while [$i -ge j]
do
j=1
while [$j -le $i]
do
echo "*\c".
 j=$((($j+1))
done
echo "\n".
i=$(($i-1))
 done
fi.
```

18. Shell script for binary to decimal conversion:-

```
echo "enter binary number"
read bin
sum=0
i=1
while [ $bin -ne 0]
do
    rem = $(expr $bin % 10)
    bin = $(expr $bin / 10)
    sum = $(expr $sum + $rem \* $i)
    i = $(expr $i \* 2)
done
echo "decimal equi is $sum".
```

Output -

enter binary num

101

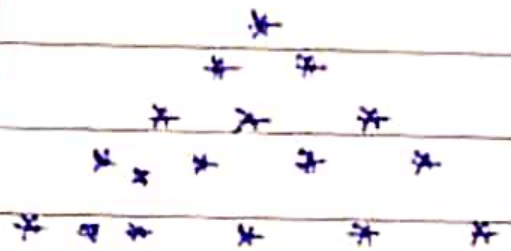
decimal equi is 5.

19. Pattern

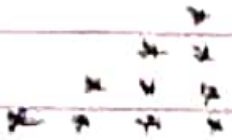
```
echo "enter n".
read n
for ((i=1; i<=n; i++))
do
  for ((j=n-i; j>=1; j--))
  do
     echo -n " "
  done
  for ((k=1; k<=i; k++))
  do
     echo -n "* "
  done
  echo
done
```

Output -

```
enter n.
5
```



20. Pattern.



```
echo "enter n"
read n.
for (( i=1 ; i<=n ; i++))
do
   for (( j= n-i ; j >=1 ; j++ ))
   do
        echo -n " "
   done
   for (( k=1 ; k<= i ; k++))
   do
      echo -n "*"
   done
        echo
done
```

Output -

```
enter n
 5
```

21. Pattern

```
* * * *
* * *
* *
*
* *
* * *
* * * *
```

```
for (( i=1 ; i<= 7 ; i++))
do
   if [ $i -le 4]
   then
      for (( j=4 ; j<=1 ; j--))
      do
         echo -n "*"
      done
   else
      for (( j=1 ; j>3 ; j--))
      do
         echo -n "*"
      done
   fi
   echo
done
```

Output-

```
* * * *
* * *
* *
*
* *
* * *
* * * *
```