Exercise No.1 Entering Commands

[mca@linux ~]$ finger

```
Login       Name      Tty      Idle  Login Time   Office     Office Phone
mca26       pts/2     Jun 17   12:00                (10.1.5.46)
mca30       pts/1     Jun 17   11:44                (10.1.5.48)
```

[mca@linux ~]$ who

```
mca30    pts/1      2015-06-17 11:44 (10.1.5.48)
mca26    pts/2      2015-06-17 12:00 (10.1.5.46)
mca45    pts/3      2015-06-17 12:02 (10.1.5.49)
```

[mca@linux ~]$ passwd

Changing password for user mca26.

Changing password for mca26

(current) UNIX password: *******

New UNIX password: *******

Retype new UNIX password: *******

passwd: all authentication tokens updated successfully.

[mca@linux ~]$ cal

```
      June 2015
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
```

[mca@linux ~]$ cal 6 2014

```
      June 2014
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
```

[mca@linux ~]$ write mca43

Hai , all the best to every one ……,

CTRL+D

[mca@linux ~]$ talk mca43

INBOX **SENT MGS**

Mca: Hai , all the best to every one Mca44:thank you,

[mca@linux ~]$ mesg y

[mca@linux ~]$ mesg n

[mca@linux ~]$ man cat

CAT(1) User Commands CAT(1)

NAME

cat - concatenate files and print on the standard output

SYNOPSIS cat [OPTION] [FILE]...

DESCRIPTION Concatenate FILE(s), or standard input, to standard output.

-A, --show-all

equivalent to –vET

[mca@linux ~]$ whoami

mca60

Exercise No:2 Files and Directory Commands

ls command

[mca@linux sitams]$ ls

regfile1 regfile2 regfile3 dir1 dir2 sym.lnk

[mca@linux sitams]$ ls -F

regfile1 regfile2 regfile3 dir1/ dir2/ sym.lnk@

[mca@linux sitams]$ ls -R

.
..
dir1 dir2 regfile1 regfile2 regfile3 sym.lnk
./dir1:
dir1file

More about Listing Files

[mca@linux sitams]$ ls –a **//includes hidden files also**

. .. regfile1 regfile2 regfile3 dir1 dir2 sym.lnk

[mca@linux sitams]$ ls –x **//lists files row wise**

abc biggest cal
dir1 dir2 dir3
hello sample1 sample2

[mca@linux sitams]$ ls –l **//lists files in long form**

total 20
-rw-rw-r-- 1 mca mca 56 Jun 16 14:16 regfile1
-rw-rw-r-- 1 mca mca 29 Jun 16 14:22 regfile2
drwxrwxr-x 2 mca mca 4096 Jun 16 14:07 dir1
-rw-rw-r--1 mca mca 0 Jun 16 14:35 regfile3

cat command

[mca@linux sitams]$ cat > file1

unix is an operating system.

[mca@linux sitams]$ cat file1

unix is an operating system.

[mca@linux sitams]$ cat >> file1

This allows to append a new line to a file called file1

[mca@linux sitams]$ cat file1

unix is an operating system.

This allows to append a new line to a file called file1

[mca@linux sitams]$ cat file2

cat command is to create a file in the current directory

[mca@linux sitams]$ cat file1 >> file2

[mca@linux sitams]$ cat file1

unix is an operating system.

This allows to append a new line to a file called file1

cat command is to create a file in the current directory

[mca@linux sitams]$ cat > file3

A new line in the file called file3

[mca@linux sitams]$ cat file2 file3 > newfile

[mca@linux sitams]$ cat newfile

cat command is to create a file in the current directory

A new line in the file called file3

[mca@linux sitams]$ file *

regfile1: ASCII text
dir2: empty
dir1: directory
regfile2: ASCII text
regfile: ASCII text

Moving around Directories

MOVING AROUND DIRECTORIES:

[mca@linux ~]$ cd sitams // to move to sitams from current directory

[mca@linux sitams]$ cd ~ //to move to home directory

[mca@linux ~]$ pwd //to view the Present Working Directory

/home/mca60

[mca@linux ~]$ cd sitams

[mca@linux sitams]$ mkdir mca

[mca@linux sitams]$ cd mca

[mca@linux sitams mca]$ pwd

/home/mca60/sitams/mca

[mca@linux sitams mca]$ cd .. // to come out of one directory

[mca@linux sitams]$ pwd

/home/mca60/sitams

Moving and Copying Files

[mca@linux sitams]$ cat file1

unix is an operating system.

[mca@linux sitams]$ mv file1 \mca **//moves the file called file1 to mca directory**
[mca@linux sitams]$ cd mca

[mca@linux sitams mca]$ cat file1

unix is an operating system.

[mca@linux sitams]$ cat file1

cat:file1:No such file or directory.

[**mca@linux sitams]$ cp file2 \mca\ file2** **// the contents of file1 is copied to file2**
//content exists in both file1 and file2

[mca@linux sitams mca]$ cat file2

cat command is to create a file in the current directory

[mca@linux sitams]$ cat file2

cat command is to create a file in the current directory

[mca@linux sitams]$ rm file3 **//the file3 is removed from current directory**

rm:remove regular file ‘file3’?y

[mca@linux sitams]$ cat file3

file3: work: No such file or directory

[mca@linux sitams]$ mkdir dir3 **//makes a new directory called dir3**

[mca@linux sitams]$ cd dir3 **//to move to dir3 directory**

[mca@linux sitams dir3]$

[mca@linux sitams]$ rmdir dir3 **//removes the directory dir3**

[mca@linux sitams]$ cd **dir3**

-bash: cd: dir3: No such file or directory

Exercise No : 3 File Search and Permission Commands

[mca@linux sitams]$ find . -name "mca*"

./mca4
./mca1
./mca3
./mca2

[mca@linux sitams]$ find . -name "mca*" > pgcourse

[mca@linux sitams]$ cat pgcourse

./mca4
./mca1
./mca3
./mca2

Changing Permissions

[mca@linux sitams]$ chmod ugo+rwx pgcourse **// Relative Permission Setting**

[mca@linux sitams]$ ls -l pgcourse

-rwxrwxrwx 1 mca mca 28 Jun 16 14:37 pgcourse

[mca@linux sitams]$ chmod 777 file1 **// obsolute Permission Setting**

-rwxrwxrwx 1 mca mca 28 Jun 16 14:37 file1

[mca@linux sitams]$ chmod 744 file2 **// all permission to user and only read permission to group and others**

[mca@linux sitams]$ ls -l file2

-rwxr--r-- 1 mca mca 28 Jun 16 14:37 file1

[mca@linux sitams]$ umask 242 **//7-2=5,7-4=3,7-2=5 i.e 535 is set to all the files which is to be created**

[mca@linux sitams]$ cat > file5

This is a new file with permissions as 5-user , 3-group and 5-others

[mca@linux sitams]$ ls –l file5

-r-x-wxr-x 1 mca mca 28 Jun 16 14:37 file5

Exercise No : 4 Unix Shell Script Programs

Program No: 1 Basic Shell Script Programs

$ vi first

```
#my first shell script
clear
echo"Knowledge is power"
```

OUTPUT

$ chmod +x first

$ ./first

Knowledge is power

$ vi second

```
#!/bin/sh
n="sun"
echo $n
```

OUTPUT

$ chmod +x second

$ ./second

sun

$ vi third

```
echo "what is your name?"
read a
echo 'hello!'$a
```

OUTPUT

S chmod +x third

$ ./thir d

what is your name?

sun

hello!sun

Program No: 2 Shell Script Variables Illustration

$ vi fouth

```
echo 'filename:' $0
echo '1st parameter:' $1
echo '2nd parameter:' $2
echo 'All arguments in single line' $@
echo 'All arguments in separate line' $*
for i in "$@";
do
echo "@'$i'";
done
echo 'Total no of arguments' $#
```

OUTPUT

$ chmod +x fourth

$ ./fourth

```
$./teste hi sun moon stars planets
filename: ./teste
1st parameter: hi
2nd parameter: sun
All arguments in single line hi sun moon stars planets
All arguments in separate line hi sun moon stars planets
@'hi'
@'sun'
@'moon'
@'stars'
@'planets'
Total no of arguments 5
```

Program No: 3 Shell Script Arrays Illustration

S vi fifth

```
#!/bin/sh
days=(mon tue wed)
```

```
echo 'First day:'${days[0]}
echo 'Second day:'${days[1]}
echo 'Third day:'${days[2]}
echo 'All week days are:'${days[*]}
```

OUTPUT

$ chmod +x fifth

$ ./ fifth

```
First day:mon
Second day:tue
Third day:wed
'All week days are:mon tue wed
```

Program No: 4 Working with Strings

1) String Length

```
s=abcABCdef
echo ${#s}
echo `expr length $s`
```

OUTPUT

```
9
9
```

2) Substring Extraction

```
s='shell script'
echo ${s:8}
echo ${s:2:6}
echo ${s:2:7}
```

OUTPUT

```
ript
ell sc
ell scr
```

3) **String replacement:**

```
s=abcABC123ABCabc
```

```
echo ${s/AB/su}
echo ${s//bcAB/sun}
```

OUTPUT

```
abcsuC123ABCabc
asunC123ABCabc
```

Program No: 5 To Perform Arithmetic Operations

$ vi Arthimetic

```
options "enter -a for addition"
echo "enter -s for substraction"
echo "enter -m for multiplication"
echo "enter -c for quotient and remainder"
echo "enter the option"
read opt
echo "enter 2 numbers"
read a
read b
case $opt in
    -a)let d=$a+$b
        echo "sum=$d"
        ;;
    -s)let d=$a-$b
        echo "difference=$d"
        ;;
    -m)let d=$a*$b
        echo "product=$d"
        ;;
    -c)let d=$a/$b
      let e=$a%$b
        echo "quotient=$d"
        echo "remainder=$d"
        ;;
     *)echo "wrong option"
        ;;
esac
```

OUTPUT

$ chmod +x Arthimetic
$./Arthimetic

```
enter -a for addition
enter -s for substraction
enter -m for multiplication
enter -c for quotient and remainder
enter the option
-a
enter 2 numbers
2
4
sum=6
```

Program No: 6 To Find Factorial of a Given Number

$ vi sixth

```
echo 'enter a value'
f=1
read n
for((i=1;i<=n;i++))
do
let f=$f*$i
done
echo 'the factorial of' $n 'is' $f
```

OUTPUT

$ chmod +x sixth

$ ./sixth

```
enter a value
5
the factorial of 5 is 120
```

Program No: 7 To Check the Given Number is Prime or Not

$ vi seventh

```
echo 'enter a number'
c=0
read n
for((i=0;i<=n;i++))
do
```

```
if test `expr $n % i` -eq 0
then
let c=$c+1
fi
done
if [ $c -eq 2];
then
echo $n 'is a prime number'
else
echo $n 'is not a prime number'
fi
```

OUTPUT

$ chmod +x seventh

$ ./seventh

```
enter a number
52
52 is not a prime number
enter a number
3
3 is not a prime number
```

Program No: 8 To Check Given Year is leap or Not

$ vi eigth

```
echo 'enter a year'
read n
if test `expr $n % 4` -eq 0
then
echo $n 'is a leap year'
else
echo $n 'is not a leap year'
fi
```

OUTPUT

$ chmod +x eigth

$ ./eigth

```
enter a number
1999
1999 is not a leap number
```

Program No: 9 To Check the Given Number is Perfect Number or Not

$ vi ninth

```
echo 'enter a number'
read n
t=$n
s=0
t=$n
for ((i=1;i<=n/2;i++))
do
if test `expr $n % $i ` -eq 0
then
let s=$s+$i
fi
done
if [ $s -eq $t ];
then
echo $t 'is a perfect number'
else
echo $t 'is not a perfect number'
fi
```

OUTPUT

$ chmod +ninth

$ ./ninth

```
enter a number
6
```

```
6 is a perfect number
enter a number
89
89 is not a perfect number
```

Program No: 10 To Check the given Number is Armstrong or not

$ vi tenth

```
echo 'enter a number'
read n
t=$n
s=0
while [ $n -gt 0 ]
do
let r=$n%10
let s=$s+$r*$r*$r
let n=$n/10
done
if [ $t -eq $s ];
then
echo $t 'is a armstrong number'
else
echo $t 'is not a armstrong number'
fi
```

OUTPUT

$ chmod +x tenth

$./tenth

```
enter a number
153
153 is a armstrong number
enter a number
167
167 is not a armstrong number
```

Program No: 11 To Check the given Number is Harshad Number or not

$ vi eleventh

```
s=0
echo 'enter a number'
read n
t=$n
a=$n
while [ $n -gt 0 ]
do
let r=$n%10
let s=$s+$r
let n=$n/10
done
if test `expr $t % $s` -eq 0
then
echo $a 'is a harshad number'
else
echo $a 'is not a harshad number'
fi
```

OUTPUT

$ chmod +x eleventh

$ ./eleventh

```
enter a number
21
21 is a harshad number
enter a number
23
23 is not a harshad number
```

Program No: 12 To Find Exponential of a Given Number

$ vi demo1

```
echo 'enter a b values'
read a b
let c=$a**$b
echo $a 'power to' $b 'is' $c
```

OUTPUT

$ chmod +x demo1

$ ./demo1

```
enter a b values
5 3
5 power to 3 is 125
```

Program No: 13 To Find sum of two numbers using Command line Arguments

$ vi demo2

```
echo 'enter a no.' $1
echo 'enter a no.' $2
echo `expr $1 + $2`
```

OUTPUT

$ chmod +x demo2

$ ./demo2

```
./h8 78 89
enter a no. 78
enter a no. 89
167
```

Program No: 14 To Find sum of two numbers using Function

$ vi demo3

```
sum()
{
echo 'enter a b values'
read a b
echo `expr $a + $b`
}
sum a b
```

OUTPUT

$ chmod +x demo3

$ ./demo3

```
enter a b values
5 6
11
```

Program No: 15 To Find Biggest of Three Numbers

$ vi demo4

```
echo 'enter 3 nos'
read a b c
if [ $a -gt $b -a $a -gt $c ];
then
echo $a 'is biggest'
elif [ $b -gt $c ];
then
echo $b 'is biggest'
else
echo $c 'is biggest'
fi
```

OUTPUT

$ chmod +x demo4

$ ./demo4

```
enter 3 nos
56 67 78
78 is biggest
```

Program No: 16 Swapping of Two Numbers using Addition and Subtraction

$ vi demo5

```
echo 'enter a b values'
read a b
let a=$a*$b
```

```
let b=$a/$b
let a=$a/$b
echo $a $b
```

OUTPUT

$ chmod +x demo5

$ ./demo5

```
enter a b values
5 6
6 5
```

Program No: 17 Shell Dcript Switch Case Illustration

$ vi demo6

```
echo 'enter 1 for addition'
echo 'enter 2 for subtraction'
echo 'enter 3 for multiplication'
echo 'enter 4 for quotient'
echo 'enter 5 for remainder'
read n
echo 'enter a b values'
read a b
case $n in
1)let d=$a+$b
echo 'sum='$d
;;
2)let d=$a-$b
echo 'sub='$d
;;
3)let d=$a*$b
echo 'mul='$d
;;
4)let d=$a/$b
echo 'div='$d
;;
```

```
5)let d=$a%$d
echo 'rem='$d
;;
*)echo 'wrong choice'
;;
esac
```

OUTPUT

$ chmod +x demo6

$ ./demo6

```
enter 1 for addition
enter 2 for subtraction
enter 3 for multiplication
enter 4 for quotient
enter 5 for remainder
3
enter a b values
4 5
mul=20
```

Program No: 18 Shell Script that displays the list of all files in the given directory

$ vi flindir

```
echo " Enter the Directory Name"
read path
echo "List of Files under " $path "are:"
ls $path
```

OUTPUT

```
[mca45@linux ~]$ chmod +x flindir
[mca45@linux ~]$ ./flindir
 Enter the Directory Name
NewDir
List of Files under  NewDir are:
```

```
file1  file2
[mca45@linux ~]$
```

Program No: 19 To Know date, User Name and Working Directory

$ vi demo8

```
echo 'enter choice:'
echo 'enter 1 for date'
echo 'enter 2 for username'
echo 'enter 3 for working directory'
read n
case $n in
1)echo
date
;;
2)echo
who mca7
;;
3)echo
pwd
;;
*)echo 'wrong choice'
;;
esac
```

OUTPUT

$ chmod +x demo8

$ ./demo8

```
enter choice:
enter 1 for date
enter 2 for username
enter 3 for working directory
1
Tue Nov  5 16:23:50 IST 2019
```

Program No: 20 To Find the Week day

$ vi demo9

```
echo 'enter the week of your choice [1-7]'
read n
case $n in
1)echo 'sunday'
;;
2)echo 'monday'
;;
3)echo 'tuesday'
;;
4)echo 'wednesday'
;;
5)echo 'thursday'
;;
6)echo 'friday'
;;
7)echo 'saturday'
;;
*)echo 'wrong choice'
;;
esac
```

OUTPUT

$ chmod +x demo9

$ ./demo9

```
enter the week of your choice [1-7]
4
Wednesday
```

Program No: 21 To Count Lines,Words, Characters in a File

$ vi count

```
echo "Enter the File Name"
```

```
read file
echo " Number of Lines in the file are :"
wc -l $file
echo "Number of Words in the File are :"
wc -w $file
echo " Number of Characters in the File are:"
wc -c $file
```

OUTPUT

```
[mca@linux ~]$ cat sample
The quick brown fox jumped over the lazy dog.
The dog drifted back to sleep and dreamed of bitting the fox
what a foolish sleepy dog
[mca@linux ~]$ chmod +x count
[mca@linux ~]$ ./count
Enter the File Name
sample
 Number of Lines in the file are :
3 sample
Number of Words in the File are :
```

Program No: 22 Shell Script that copies multiple Files to a Directory

$ vi mulindir

```
i=1
echo "Enter New Directory Name"
read dir
mkdir $dir
echo "Enter the Number of Files to be Added"
read n
while [ $i -le $n ]
do
    echo "Enter the file Name"
    read filename
    cp $filename $dir
    let i=$i+1
```

```
done
```

OUTPUT

```
 [mca45@linux ~]$ chmod +x mulindir
[mca45@linux ~]$ ./mulindir
Enter New Directory Name
NewDir
Enter the Number of Files to be Added
2
Enter the file Name
file1
Enter the file Name
file2
[mca45@linux ~]$ cd NewDir
[mca45@linux NewDir]$ ls
file1  file2
[mca45@linux NewDir]$
```

Program No: 23 Shell Script TO Generate a Multiplication Table

$ vi multable

```
echo "Enter Which table you want"
read n
for i in 1 2 3 4 5 6 7 8 9 10
do
    let k=$n*$i
    echo "$n X $i = $k"
done
```

OUTPUT

```
[mca45@linux ~]$ chmod +x multable
[mca45@linux ~]$ ./multable
Enter Which table you want
3
3 X 1 = 3
3 X 2 = 6
```

```
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
[mca45@linux ~]$
```

Exercise No : 5 Pattern Matching and file compression

grep commands

[mca@linux ~]$ cat > sample

This is unix lab

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ cat sample

This is unix lab

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ grep commands sample **// one word as pattern**

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ grep " unix lab" sample **// pharse or multiple words as pattern**

This is unix lab

[mca@linux ~]$ grep lab * **// wild card in grep command**

Sample: This is unix lab

First : This is the first unix file with vi editor

[mca@linux ~]$ grep 'Th\(i\|u\)' sample

This is unix lab

[mca@linux ~]$ grep -i UNIX sample **// -i to ignore the case**

This is unix lab

[mca@linux ~]$ grep -v unix sample **// displays lines that does not contain unix pattern**

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ grep -n unix sample // displays matched lines with line numbers

1:This is unix lab

[mca@linux ~]$ grep -l unix sample //displays the filenames which contains the pattern unix

Sample

Demo

[mca@linux ~]$ grep -c unix sample// displays number of lines that contains the pattern “unix” in sample file

1

fgrep command

[mca@linux ~]$ fgrep "searches pattern" sample \\ only fixed strings can be used as target

The commands searches pattern in a file

egrep command

[mca@linux ~]$ egrep "unix|file|commands" sample

This is unix lab

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ grep 'Th(i|u)' sample \\ No Need to escape the special characters in egrep

This is unix lab

gzip command

[mca@linux ~]$ gzip -v sample

sample: 24.4% -- replaced with sample.gz

[mca@linux ~]$ gunzip -v sample

sample.gz: 24.4% -- replaced with sample

Exercise No : 6 Counting Lines, words and size of the file

wc and nl commands

[mca@linux ~]$ wc sample

3 15 79 sample

[mca@linux ~]$ wc -l sample

3 sample

[mca@linux ~]$ wc -c sample

79 sample

[mca@linux ~]$ wc -w sample

15 sample

[mca@linux ~]$ wc -L sample

39 sample

[mca@linux ~]$ grep commands sample | wc -l

2

[mca@linux ~]$ nl sample

1 This is unix lab

2 The commands searches pattern in a file

3 We are doing commands

Exercise No : 7 Working with columns or fields ,sort and uniq commands

cut command to extract specific fields and characters

$ cat > students

```
01  hari    MCA  80  88  89  77  67
02  siva    MCA  90  88  89  80  89
03  ramu    MCA  89  77  78  79  67
04  raghu   MCA  90  95  77  88  67
05  dinesh  MCA  98  90  78  67  77
```

$ cut -c4 students **// displays 4th column of the file**

```
h
s
r
r
d
```

$ cut -c2 students **//displays 2nd column of the file**

```
1
2
3
4
5
```

$ cut -c4,7 students **//displays 4th and 7th columns of the file**

```
hi
sa
ru
rh
de
```

$ cut -c4-7 students **//displays from 4th to 7th columns of the file**

```
hari
siva
ramu
ragh
dine
```

$ cut -c-7 students **//displays from 1st to 7th characters of the file**

```
01  hari
02  siva
03  ramu
04  ragh
05  dine
```

$ cut -c7- students **//displays from 7th to last column of the file**

```
i     MCA  80  88  89  77  67
a     MCA  90  88  89  80  89
u     MCA  89  77  78  79  67
hu    MCA  90  95  77  88  67
esh   MCA  98  90  78  67  77
```

$ cut -f2 students **//displays 2nd field of the file**

```
hari
```

```
        siva
        ramu
        raghu
        dinesh

$ cut -f1 students                          //displays 1st field of the file
        01
        02
        03
        04
        05
$ cut -f2,3 students                        //displays 2nd and 3rd fields of the file
        hari    MCA
        siva    MCA
        ramu    MCA
        raghu   MCA
        dinesh  MCA
$ cut -f2-4 students                        //displays 2nd to 4th  fields of the file
        hari    MCA     80
        siva    MCA     90
        ramu    MCA     89
        raghu   MCA     90
        dinesh  MCA     98+-961
$ cut -f1-3 students                        //displays from 1st to 3rd fields
        01      hari    MCA
        02      siva    MCA
        03      ramu    MCA
        04      raghu   MCA
        05      dinesh  MCA
$ cut -f-4 students                         //displays from 1st to 4th  fields
        01      hari    MCA     80
        02      siva    MCA     90
        03      ramu    MCA     89
        04      raghu   MCA     90
        05      dinesh  MCA     98
```

colrm command to delete specific characters

```
$ colrm 9 10  students > newfile            //removes 9th and 10th characters

$ cat  newfile
        01      hari    A   80  88  89  77  67
        02      siva    A   90  88  89  80  89
        03      ramu    A   89  77  78  79  67
        04      raghu   A   90  95  77  88  67
        05      dinesh  A   98  90  78  67  77
```

paste command to gue files line by line

[mca@linux ~]$ cat > states

alabama
alaska
arizona
arkansas
California

[mca@linux ~]$ cat >state_abbr

al
ak
az
ar
ca

[mca@linux ~]$ paste states state_abbr >states.imp

[mca@linux ~]$ cat states.imp

Alabama al
alaska ak
arizona az
arkansas ar
california ca

join command to join files based on key field

[mca@linux ~]$ cat > merch

63A457 mans gold watch
73B312 garnet ring
82B119 sapphire pendant

[mca@linux ~]$ cat > costs

63A457 125.50
73B312 255.00
82B119 534.75

[mca@linux ~]$ join merch costs

63A457 mans gold watch 125.50
73B312 garnet ring 255.00
82B119 sapphire pendant 534.75

Sort commands

[mca@linux ~]$ cat > names

```
01    Uma
05    Jennifer
02    ram
04    priya
06    raj
03    Shreya
03    Shreya
05    Jennifer
```

[mca@linux ~]$ sort names **\\ Sorts based on first Column**

```
01    Uma
02    ram
03    Shreya
03    Shreya
04    priya
05    Jennifer
05    Jennifer
06    raj
```

[mca@linux ~]$ cat > names1

```
nani
Nani
Arjun
Jeniffer
```

[mca@linux ~]$ sort -f names1 **\\ sorts by ignoring the case**

```
Arjun
Jeniffer
nani
Nani
```

[mca@linux ~]$ cat > student

```
smith   100
jimmer  103
john    102
Shreya  104
```

[mca@linux ~]$ sort -n student **\\sorts based on numerical values**

```
jimmer  103
john    102
Shreya  104
smith   100
```

[mca@linux ~]$ sort -r names **\\sorts based on first column in reverse order**

```
06    raj
05    Jennifer
05    Jennifer
04    priya
03    Shreya
03    Shreya
02    ram
01    Uma
```

[mca@linux ~]$ sort -u names **\\sorts by ignoring the duplicate values**

```
01    Uma
02    ram
```

```
        03      Shreya
        04      priya
        05      Jennifer
        06      raj
[mca@linux ~]$ sort -k2 names          \\sorts based on specific field (i.e 2nd field)
        05      Jennifer
        05      Jennifer
        04      priya
        06      raj
        02      ram
        03      Shreya
        03      Shreya
        01      Uma
```

uniq commands

```
[mca@linux ~]$cat kt.txt
        I love music.
        I love music.
        I love music.

        I love music of Kartik.
        I love music of Kartik.

        Thanks.
[mca@linux ~]$uniq -c kt.txt      \\at the starting of each line its repeated number is displayed
        3 I love music.
        2 I love music of Kartik.
        1 Thanks.
[mca@linux ~]$uniq -d kt.txt      \\it only displayed one  duplicate line per group
        I love music.
        I love music of Kartik.
[mca@linux ~]$uniq -D kt.txt      \\ all the duplicate lines are displayed
        I love music.
        I love music.
        I love music.
        I love music of Kartik.
        I love music of Kartik.
 [mca@linux ~]$uniq -u kt.txt     \\only unique lines are displayed
          Thanks.
```

Exercise No : 8 File Comparison and Editing Commands

cmp, comm, diff and patch commands

[mca@linux ~]$ cat > notes

Nate,
heres the first draft of the plan
i think it needs more work

[mca@linux ~]$ cat > notes.more

Nate,
heres the first draft of the new plan
i think it needs more work
let me know what you think

[mca@linux ~]$ cmp notes notes.more

notes notes.more differ: byte 36, line 2

[mca@linux ~]$ cat > cities1

newyork
palo alto
san fransico
seattle

[mca@linux ~]$ cat > citites2

palo alto
san fransico
santa monica
seattle

[mca@linux ~]$ comm cities1 cities2

newyork
palo alto
san fransico
santa monica
seattle

[mca@linux ~]$ diff notes notes.more

2c2
< heres the first draft of the plan

> heres the first draft of the new plan
3a4
> let me know what you think

[mca@linux ~]$ diff notes notes.more > difference

[mca@linux ~]$ cat difference

2c2
< heres the first draft of the plan---

> heres the first draft of the new plan
3a4
> let me know what you think

[mca@linux ~]$ patch notes difference
patching file notes

editing and formatting files

[mca@linux ~]$ cat >names
Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net
[mca@linux ~]$ pr names
2015-06-16 15:08 names Page 1
Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net
[mca@linux ~]$ cat names
Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net
[mca@linux ~]$ cat > sample1
this is an example of
a short
file
that contains lines of varying width
[mca@linux ~]$ fmt -w 16 sample1
this is an
example of a
short file that
contains lines
of varying
width
[mca@linux ~]$ cat > newfile
bin
robin
dan
[mca@linux ~]$ tr '[a-z]' '[A-Z]' < newfile
BIN
ROBIN
DAN
[mca24@linux ~]$ cat textfile
spell is a unix command alloes to chech yje spelling in a file.
[mca24@linux ~]$ spell textfile
alloes
chech
unix

Exercise No : 9 Data , bc and dc commands

Date command

List of Format specifiers used with date command:

%D: Display date as mm/dd/yy.
%d: Display the day of the month (01 to 31).
%a: Displays the abbreviated name for weekdays (Sun to Sat).
%A: Displays full weekdays (Sunday to Saturday).
%h: Displays abbreviated month name (Jan to Dec).
%b: Displays abbreviated month name (Jan to Dec).
%B: Displays full month name(January to December).
%m: Displays the month of year (01 to 12).
%y: Displays last two digits of the year(00 to 99).
%Y: Display four-digit year.
%T: Display the time in 24 hour format as HH:MM:SS.
%H: Display the hour.
%M: Display the minute.
%S: Display the seconds.

Syntax:
$date +%[format-option]

Command:
$date "+%D"
Output:
10/11/17
Command:
$date "+%D %T"
Output:
10/11/17 16:13:27
Command:
$date "+%Y-%m-%d"
Output:
2017-10-11
Command:
$date "+%Y/%m/%d"

Output:
2017/10/11
Command:
$date "+%A %B %d %T %y"
Output:
Thursday October 07:54:29 17

bc command : The **bc** command is both a calculator and a mini-language for writing mathematical programs. It provides all of the standard arithmetic operations, as well as a set of control statements and user definable functions.

bc operators and Functions

Symbol	Operation	Symbol	Operation
+	Addition	sqrt(*x*)	Square root
–	Subtraction	scale=*n*	Set scale
/	Division	ibase=*n*	Set input base
*	Multiplication	obase=*n*	Set output base
%	Remainder	define f(x)	Define function
A	Exponentiation	for, if, while	Control statements
()	Grouping	quit	Exit **bc**

$ bc
32+17
49
sqrt (49)
7
Quit

$ bc
scale=4 \\scale is to specify number of decimal places to be preserved
sqrt (2)
1.4142

Control Statement Illustration

```
$ bc
for(i=1;i<=4;i=i+1) i^2
```

output :

```
1
4
9
16
```

Function

```
define pyth(a,b)
{
c=a^2+b^2
return(sqrt(c))
}
pyth(3,4)
```

output :

```
5
```

Reading Functions from Files

```
$ bc lib.bc                    // Reading f function from the file
lib.bc
f (5)
```

output :

```
120
```

dc command: The **dc** command gives you a calculator that uses the *Reverse Polish Notation (RPN)* method of entering data and operations. This approach is based on the concept of a *stack* that contains the data you enter, and *operators* that act on the numbers at the top of the stack

dc operators

Symbol	Operation	Symbol	Operation
+	Addition	P	Print top item on stack
–	Subtraction	d	Duplicate top item on stack
/	Division	r	Reverse top two items on stack
*	Multiplication	f	Print entire stack
%	Remainder	c	Clear stack
^	Exponentiation	s*x*	Save to memory register *x*
v	Square root	l*x*	Load from register *x*
k	Set scale	I	Set input base
q or Q	Exit program	o	Set output base

```
$ dc
      32 64+p
      96
      q

$ dc
      4 k 2vp // k means set scale to 4 ,v means square root and p means to print
      1.4142

$ dc
      1 5+ 3 4+ * 6/ 2^
      49
```

Exercise No : 10 awk and sed

How awk Works

[mca@linux ~]$ cat > inventory

awk program was originally developed by Aho,kernighan and Weinberger in 1977.

It is also called as pattern scanning language.

[mca@linux ~]$ awk '/scanning/{print}' inventory **//displays lines that contains the //"scanning" pattern**

It is also called as pattern scanning language.

Default Patterns and Actions

[mca@linux ~]$ cat contacts

Ben	IN	650-333-4321
Dan	AK	907-671-4321
Morissa	NJ	732-741-2431
Robin	CA	650-273-1034

[mca@linux ~]$ awk '{print $1}' inventory **//default pattern**

Ben
Dan
Morissa
Robin

[mca@linux ~]$ awk '/Aho/' inventory **//default action**

Awk program was originally developed by Aho,kernighan and Weinberger in 1977.

Working with Fields

[mca@linux ~]$ cat > contacts

Ben	IN	650-333-4321
Dan	AK	907-671-4321
Morissa	NJ	732-741-2431
Robin	CA	650-273-1034

[mca@linux ~]$ awk '/650-/{print $2}' contacts **//displays 2nd fields that matches the pattern**

IN
CA

Using Standard Input and Output

[mca@linux ~]$ awk '{print}' inventory > standard

[mca@linux ~]$ cat standard

Awk program was originally developed by Aho,kernighan and Weinberger in 1977.

It is also called as pattern scanning language.

Running awk Program from a file

[mca22@linux ~]$ cat > fmtdemo **//awk commands in file fmtdemo**

```
/Mor+/{print $1}
```

[mca@linux ~]$ awk -f fmtdemo contacts **//executing fmtdemo program**

Morissa

Multiline Programs

[mca@linux ~]$ cat > numberline

```
{
n=n+1
print n " " $0
}
```

[mca@linux ~]$ awk -f numberline contacts //Executing numberline program on contacts

```
1 Ben          IN    650-333-4321
2 Dan          AK    907-671-4321
3 Morissa      NJ    732-741-2431
4 Robin        CA    650-273-1034
```

Specifying Patterns

REGULAR EXPRESSION :

[mca@linux ~]$ cat > stationary1

```
pencil    100   2     3
markers   50    20    22
pens      200   10    12
notes     200   30    34
```

[mca@linux ~]$ awk '/pe*/{print $0}' stationary1

```
pencil  100   2     3
pens    200   10    12
```

Comparision Operators

[mca@linux ~]$ awk '$3 >10 {print $0}' stationary1

markers 50 20 22

notes 200 30 34

HOW SED WORKS

[mca@linux ~]$ cat > data1

Sed is an abbrevation for stream editor like awk,
it can do complex pattern matching and editing
on a stream of characters,although it does not
have all the powerful programming capablities
of awk.

[mca@linux ~]$ cat data1

Sed is an abbrevation for stream editor like awk,
it can do complex pattern matching and editing
on a stream of characters,although it does not
have all the powerful programming capablities
of awk.

Selecting Lines

[mca@linux ~]$ sed '1d' data **// deletes first line**

it can do complex pattern matching and editing
on a stream of characters,although it does not
have all the powerful programming capablities
of awk.

Regular Expression

[mca@linux ~]$ sed '/awk/d' data **//deletes lines that contains “awk”**

it can do complex pattern matching and editing
on a stream of characters,although it does not
have all the powerful programming capablities

By Line Number

[mca@linux ~]$ sed '1,3d' data **//removes 1st and 3rd lines from data file**

have all the powerful programming capablities

of awk.

Editing Command

[mca@linux ~]$ sed -n '1,2p' data

Sed is an abbrevation for stream editor like awk,

it can do complex pattern matching and editing

Rep lacing Strings

[mca@linux ~]$ sed 's/awk/AWK/g' data

Sed is an abbrevation for stream editor like AWK,

it can do complex pattern matching and editing

on a stream of characters, although it does not

have all the powerful programming capablities

of AWK.