PRACTICAL-5

Aim: Filter commands

* Sort:

=> It is used to sort the lines in a text file.

=> sort by a-z, it also be used to sort numerically.

Syntax: sort [options] filename

Options:

-b ignore space at beginning of the line
-c check whether input is sorted
-r sorts in reverse order
-u if line is duplicated only display once
-n compare accor. to string numerical value
-nr To ~~sto~~ sort a file with numeric date in reverse
-k sorting a table on basis of ~~on~~ any column

Output: aditi@DESKTOP-J7E31H6:~$ sort xyz

bycc
Have a good day
hello world
Kem cho?
How are you?
Good morning.

* Paste:

Paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

Syntax: Paste [options] file
Options:
-d specify of a list of delimitors
-s paste one file at a time instead of in parallel.
Output: 20cc39@localhost:~$ paste

* Grep:
=> Scans a file for the occurance of a pattern & can display the selected pattern, the line numbers in which they are found or the filenames where the pattern occurs.
=> Grep can display the lines not contain the pattern.
Syntax: grep [options] pattern [file]
Options:
-l -I o
-c Print a count of matching lines for each i/p
-n Display the matched lines & their no.
-v Invert the sense of ~~mod~~ matching to select non-matching lines.
Output: 20cc39@localhost:~$ grep good xyz
Good morning

* Uniq:

=> Uniq remove duplicate records, simply fetches one copy of each records & write into the std. output.

=> The line should be already sorted before using Uniq command.

Syntax: Uniq filename

Options:

-c precede each output line with a count of the no. of times the line occured in.

-d suppress the writing of line that are not repeated in I/p.

-D print all duplicate line

-f Avoid comparing first N fields

-i ignore case when comparing

-s Avoid comparing first N characters

Output: soces9@localhost:~$ uniq xyz

Hello world

How are you?

Good morning

Have a good day

Byee

* Head:

=> Head command is used to display the first ten lines of a file, and also specifies how many lines to display.

Syntax: Head filename

-n to specify how many lines you want to display

Output: 20ce39@localhost: ~ $ head XYZ
Hello world
Good morning
How are you?
Have a good day
Byee

* nl:
nl command numbers the line in a file
Syntax: nl filename
Output: 20ce39@localhost: ~ $ nl XYZ
1 Hello world
2 Good morning
3 How are you?
4 Have a good day
5 Byee

* tail:
tail command is used to display the last or bottom part of the file. By default it displays last ~~to~~ 10 lines of a file.
Syntax: tail filename
-n to specify how many lines you want to display
Output: 20ce39@localhost: ~ $ tail XYZ
Hello world
Good morning
How are you?
Have a good day
Keep cho?
byee

* tr: Translating character.

=> This command takes it i/p only from it std. i/p, it doesn't take filename as input. It translate each character in A to its map counter part in B.

cat filename | tr "[A-Z]" "[A-Z]"

echo "name" | tr [" "] '/t' (for line)

Output: zoce39@localhost: ~$ cat xyz
| tr "[A-Z]" "[A-Z]"
Hello world
Good morning
Kem cho?
How are you?
Have a good day
byee

zoce39@localhost: ~$ echo "day"
| tr [" "] '/t'
day

* Cut:

=> cut command used to cut out selected fields of each line of a file. The cut command uses delimiters to determine where to split fields.

Syntax: cut [options] filename

Options:

-c The list following -c specifies char. position

-d The char. following -d is the field delimiter

-f select only these fields on each line
-b select only the bytes from each lines as specified in list.

Output: soce39@localhost:~$ cut -c 3-3,1,2
llo wor
Morning i
m cho1
iams
ve u go
ec

PRACTICAL 6

Aim: To study how to use vi editor.

=> Vi editor is a very Powerful tool & has a very extensive built-in manual, which can be activated using the ':help' command when the program is started.

=> Vi can operate in two modes
1) command mode
2) Insert mode

=> We use 'ESC key' to get into command mode & 'i' key to get into INSERT mode.

* Basic Commands:

1) Moving through the text
   h -> Move the cursor to the left
   l -> Move the cursor to right
   k -> move up
   j -> Move down

2) Basic operation:
   dd -> Delete the line on which the cursor is position.
   dnd -> Delete n lines starting from cursor position.
   :w -> Save the file
   :q -> Exit the editor
   :q! -> Forces the exit whom you want to quit a file with unsaved change
   :wq -> save & exist.

:wq! → overrides read-only permissions.

/a string → Search the string in file & position the cursor on first match ^ position

yy → Copy a block of text

n p → paste it n times.

PRACTICAL 7

Aim: 1) Write a shell script to find factorial of given number n.
2) Write a shell script to finds Whether given number is prime or not.

1) Shell script to find factorial of given number n.

=>

```
echo "Total no of factorial wants"
read fact

ans=1
counter=0
while [ $fact -ne $counter]
do
        counter=`expr $counter + 1`.
        ans=`expr $ans \* $counter`
done
echo "Total of factorial is $ans"
```

Output:

```
Total no of factorial wants
5
Total of factorial is 120
```

2) Shell script to finds Whether given number is prime or not.

=>

```
echo -n "Enter a number: "
read num
```

```
i=2

while [ $i -lt $num ]
do
   if [ `expr $num % $i` -eq 0 ]
   then
        echo " $num is not a prime number"
        echo "Since it is divisible by $i"
        exit
    fi
    i=`expr $i+1`
done
echo " $num is a prime number"
```

Output:

```
Enter a number : 1879
1879 is a prime number
[root @ venu] # . /prime1.sh
Enter a number : 119
119 is not a prime number
Since it is divisible by 7
```

Using here string you can supply input non interactively.

PRACTICAL 8

Aim: 1) Write a shell script to check entered string is palindrome or not.
2) Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3, 5, 8, ...

1) Shell Script to check entered string is palindrome or not.

=>

```
clear
echo "Enter a string to be entered:"
read str
echo
len=`echo $str | wc -c`
len=`expr $len -1`
i=1
j=`expr $len / 2`
while test $i -le $j
do
k=`echo $str | cut -c $i`
l=`echo $str | cut -c $len`
if test $k != $l
then
echo "string is not palindrome"
exist
fi
i=`expr $i + 1`
len=`expr $len -1`
done
echo "String is palindrome"
```

2) First n Fibonacci numbers like: 1, 1, 2, 3, 5, 8, ...

=>

```
if [ $# -eq 1 ]
then
    Num=$1
else
    echo -n "Enter a number : "
    read Num
Fi
f1=0
f2=1
echo " The Fibonacci sequence for the
      number $Num is : "
for (( i=0 ; i<=Num ; i++ ))
do
    echo -n " $f1 "
    fn=$((f1+f2))
    f1=$f2
    f2=$fn
done
```

Output:

```
# sh fibo_intenative.sh 18
 The fibonacci sequence for the number
 18 is:
0 1 1 2 3 5 8 13 21 34 55 89 144 233
377 610 987 1597 2584
```

PRACTICAL 9

Aim: Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate & display total marks, percentage & Cluss obtained by the student.

=>
```
echo Enter the Mark1
read m1
echo Enter the Mark2
read m2
echo Enter the Mark3
read m3

tot=$(expr $m1 + $m2 + $m3)
echo "Total : $tot"
avg=$(expr $tot / 3)
echo "Average : $avg"
if [ $m1 -ge 35 ] && [ $m2 -ge 35 ]
 && [ $m3 -ge 35 ]
then
echo "Result : Pass"

if [ $avg -ge 90 ]
then
   echo "Grade : S"
elif [ $avg -ge 80 ]
then
   echo "Grade : A"
elif [ $avg -ge 70 ]
then
   echo "Grade : B"
```

vision

```
elif [$avg -ge 60]
then
    echo "Grade : C"
elif [$avg -ge 50]
then
    echo "Grade : D"
elif [$avg -ge 35]
then
    echo "Grade : E"
fi
else
   echo "Result : Fail"
fi
```

PRACTICAL 10

Aim: 1) Write a shell script to validate the entered date.
2) Write a menu driven shell script to make calculator using switch case.

1) Validate the entered date.

=>

```
usrdate = $1
date -d "$usrdate" > /dev/null 2>&1
if [ $? -eq 0 ];
then
    echo "Date $usrdate was valid"
else
    echo "Date $usrdate was invalid"
fi
```

2) A menu driven shell script to make calculator using switch case.

=>

```
while
do
 clear
 echo "******* Task performing script
      *******"
 echo "1. task 1"
 echo
 echo "2. task 2"
 echo
 echo "3. task 3"
 echo
```

```
echo "4. Exit"
echo
echo -n "Please enter any option or
  options between [1 -4]"
read opt
case $opt in
1)
 echo "Performing task 1...."
 for i in `cat test`
  do
  .......
  .......
  done;;
2)
  echo "performing task 2...."
  for i in `cat test`
  do
  .......
  .......
  done;;
3)
  echo "Performing task 3......"
  for i in `cat test`
  do
  .......
  .......
  done;;
4)
  echo "Bye $USER";
  exit 1;;
*)
```

```
        echo "$opt is an invalid option";
        echo "Press [entrc] key to continue...";
        read enterkey;;
        esac
done
```