ExNo:01

FILE MANIPULATION COMMANDS

AIM:

To write a shell script to stimulate the basic Linux commands: rm, cmp, cat, cp, mv, wc, split, diff.

ALGORITHM:

Step1: Start the process.

Step2: In the shell script prompt window perform the required commands.

Step3: Use ls command to list the files and directories in the current directory.

Step4: Copy the contents of the file1 to file3, use cp command.

Step5: Use cat command to view the copied content in the file3.

Step6: Use mv command to move the contents in file2 to file4. And view the content in file4.

Step7 : Use wc command to display count of the lines, words, character in the file file4.txt

Step8: Use wc -l, wc -w, wc -c to display the count of lines, word, character respectively.

Step9: Use the split command to split the contents in the file4.

Step10: Use cmp command to compare the contents file1 and file4

Step11: Using diff command between file1.txt file4.txt

Step12: Stop the process.

SHELL SCRIPT:

```
echo “ ls command”

ls

echo "Copying content from file1.txt to file3.txt"

cp file1.txt file3.txt

echo "Displaying contents in file3.txt"

cat file3.txt

echo "Moving contents from file2.txt to file4.txt"

mv file2.txt file4.txt
```

```
echo "Displaying content in

file4.txt" catfile4.txt

echo "Display count of the lines, words, character in the file file4.txt"

wcfile4.txt

echo "Display the number of lines in file4.txt"

wc -l file4.txt

echo "Display the number of words in file4.txt"

wc -w file4.txt

echo "Display the number of character in file4.txt"

wc -c file4.txt

echo "Splitting the file4.txt by 3 lines in each"

split -3 file4.txt

echo "Listing the files"

ls

echo "Comparing the files file1.txt file4.txt"

cmp file1.txt file4.txt

echo "Using diff command between file1.txt file4.txt"

diff file1.txt file4.txt
```

OUTPUT:

ls command

content.txt file3.txt new.txt program1.sh program2.sh program4.sh program5.sh progra

m7.sh xab

file1.txt file4.txt passwd program1.txt program3.sh program4.txt program6.sh

xaaxac

Copying content from file1.txt to file3.txt

Displaying contents in file3.txt

a

b

c

d

e

f

g

The appended content

Moving contents from file2.txt to file4.txt

Displaying content in file4.txt

sun

mon

tue

wed

thuf

ri

sat

Display count of the lines, words, character in the file file4.txt

7 7 28 file4.txt

Display the number of lines in file4.txt

7file4.txt

Display the number of words in

file4.txt 7file4.txt

Display the number of character in file4.txt

28 file4.txt

Splitting the file4.txt by 3 lines in each

Listing the files

content.txt file3.txt new.txt program1.sh program2.sh program4.sh program5.sh progra

m7.sh xab

file1.txt file4.txt passwd program1.txt program3.sh program4.txt program6.sh

xaaxac

Comparing the files file1.txt file4.txt

file1.txt file4.txt differ: byte 1, line 1

Using diff command between file1.txt file4.txt

1,8c1,7

<a

<b

<c

<d

<e

<f

< g

< The appended content

\ No newline at end of file

> sun

> mon

> tue

> wed

> thu

> fri

> sat

ExNo:02

SYSTEM CONFIGURATION COMMANDS

AIM:

To write a shell script to implement the user and system information by commands.

ALORITHM:

Step1: Start the process.

Step2: In the shell script prompt window perform the following command: use LOGNAME to check login name of the user, SHELL to check the following shell information.

Step3: Type OSTYPE command to check the OS type of the Linux OS.

Step4: Type path to check the path for particular directories.

Step5: Type pwd command to view the present working directory.

Step6: lscpu command to check the CPU information.

Step7: Stop the process.

SHELL SCRIPT:

```
echo "User Name: " $USER

echo "Login Name: " $LOGNAME

echo "Current Shell: " $SHELL

echo "List of Shells: "

chsh -l

echo "Home Directory: " $HOME

echo "Our PC OS is: "$OSTYPE

echo "Current Path" $PATH

echo "Current Directory: "

pwd

echo "List of Logged Users"

who|wc -l

echo "System Configuration (or) PC Configuration: "
```

lscpu

echo "Free Memory Space: "

free –m

echo “Showing all Memory Information:”

cat/proc/meminfo

OUTPUT:
User Name: 1926ka38

Login Name: 1926ka38

Current Shell: /bin/bash

List of Shells:

/bin/sh

/bin/bash

/sbin/nologin

/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh

/bin/zsh

Home Directory: /home/1926ka38
Our PC OS is: linux-gnu
Current Path:
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/1926ka38 /.composer/vendor/bin:/us
r/lib64/openm
pi/bin:/usr/bin:/home/1926ka38 /.local/bin:/home/1926ka38 /bin
Current Directory:
/home/1926ka38
/linuxLab List of Logged
users:
43
System Configuration (or) PCConfiguration:
Architecture: x86_64
CPUop-mode(s): 32-bit,
64-bit Byte Order: LittleEndian
CPU(s): 2
On-line CPU(s) list: 0,1
Thread(s)percore: 1
Core(s)persocket: 2
Socket(s): 1
Free memory space:

	total	used	free	shared	buff/cache	available
Mem:	3683	547	2634	33	501	2759
Swap:	3967	0	3967			

ExNo:03

IMPLEMENTION OF PIPE, REDIRECTION AND TEE COMMANDS

AIM:

To write a shell script to implement the following pipe, redirection and tee commands.

ALGORITHM:

Step1: Start the process.

Step2: Use (|) pipe command to link one command with another (or) one operation.

Step3: Use (>>) command to transfer the file from one part into another.

Step4: Use more command to check the information on the screen.

Step5: Display the result on the screen.

Step6: Save the process.

Step7: End the process.

SHELL SCRIPT:

```
echo "Exploring Pipes and Redirection Commands"

echo "Using pipe command:"

ls –l | wc

echo "Using tee command:"

ls –l | wc | tee new.txt

echo "Content in new.txt"

cat new.txt

echo "Using redirection command:"

echo "Content in file1.txt before redirecting:"

cat file1.txt

echo "Redirecting commands is tobeexecuted.     "

cat >> file1.txt

echo "The content in file1.txt after
```

appending:" catfile1.txt

OUTPUT:

Exploring pipes and Redirection Commands

Using pipe command:

19 164 1107

Using tee command:

19 164 1107

Content in new.txt

19 164 1107

Using redirection command:

Content in file1.txt before redirecting:

a

b

c

d

e

f

g

The appended content

Redirecting commands is to be executed....

this is the appended text
The content in file1.txt after appending:
a
b
c
d
e
f
g
The appended content
this is the appended
text

ExNo:04

SHELL SCRIPT FOR DISPLAYING DATE, USERNAME AND LISTING THE FILES AND DIRECTORIES

AIM:
To write a shell script to display the current date, username and list of files and directories by getting users choice.

ALGORITHM:
Step 1: Start the process.

Step 2: Use a case statement for performing a different action into a single prompt.

Step 3: Declare case variable and case command for the program.

Step 4: If the choice is 1 then the current date will be displayed on the screen.

Step 5: If the choice is 2 than username will be shown if the choice is 3 then file can be listed along with the directories.

Step 6: If name of the choice is met finally default case get executed on the screen.

Step 7: Stop the process

SHELL SCRIPT:

```
echo "1.Current date:"

echo "2.Your user name:"

echo "3.List files and directories"

read option

case ${option} in

1)

echo "Current date is :" $(date);;

2)

echo "Your user name is: "$(whoami);;

3)
```

```
echo "To list out all files and directories:"$(ls);;

*)

echo "Invalid Option"

esac
```

OUTPUT:

1. CurrentDate:

2. Your username:

3. List files and

directories 1

Current date is: Fri Feb 26 15:58:15 IST 2021

ExNo:5

IMPLEMENTATION OF FILTER COMMANDS

AIM:

To write a shell script to implement filter commands.

ALGORITHM:

Step 1: Start the process.

Step 2: Create a file using vi editor.

Step 3: Copy the file /etc/passwd to passwd file.

Step 4: To display the lines containing the word root use grep -n "root" passwd.

Step 5: To display the no.of lines containing the word root use grep -c "root"passwd.

Step 6: To display all lines, words, characters in passwd file use wcpasswd.

Step 7: To display all the lines that do not match with the line root use grep -v"root "passwd.

Step 8: To replace ":"with "*"in the file passwd use tr":" "*"<passwd.

Step 9: To display the first column of the file passwd use cut -d ':' -f1 passwd.

Step 10: The output will be displayed on the screen.

Step 11: Stop the process.

SHELL SCRIPT:

```
echo "Filter commands"

cp /etc/passwd passwd

echo "Display the lines containing the word root"

grep -n "root" passwd | more

echo "Display the count of lines that is containing the word root"

grep -c "root" passwd

echo "Display the count of lines that dont match with the line root"

grep -v "root" passwd | more

echo "Display the no. of lines, character and words in passwd file"

wc passwd
```

```
echo "Replace ":" with "*" in the passwd file"

tr ':' '*' < passwd | more

echo "Display first column of the passwd file"

cut -d ':' -f1 passwd
```

OUTPUT:

```
[1922kc62@Kgcaslinux lab]$sh program5.sh

Filter commands

Display the lines containing the word root

1:root:x:0:0:root:/root:/bin/bash

10:operator:x:11:0:operator:/root:/sbin/nologin

Display the count of lines that is containing the word root

2

Display the count of lines that dont match with the line root

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

Display the no. of lines, character and words in passwd file
```

1626 1670 76018 passwd

Replace : with * in the passwd file

root*x*0*0*root*/root*/bin/bash

bin*x*1*1*bin*/bin*/sbin/nologi

n

daemon*x*2*2*daemon*/sbin*/sbin/nologinadm*

x*3*4*adm*/var/adm*/sbin/nologinlp*x*4*7*lp*

/var/spool/lpd*/sbin/nologin

sync*x*5*0*sync*/sbin*/bin/sync

shutdown*x*6*0*shutdown*/sbin*/sbin/shutdown

ExNo:06

DELETE THE FILE WHICH HAS SIZE AS ZERO FILE SIZE

AIM:

To write a shell script to remove the file which has size as zero bytes.

ALGORITHM:

Step 1: Start the process.

Step 2: create a file using vi editor.

Step 3: Get a file name as input from the user.

Step 4: Usetheif else statement check thecondition.

Step 5: If the file exists then check the size of the file.

Step 6: If the size of file is zero then remove the file.

Step 7: Remove the file using rm command .if not leave it as it is.

Step 8: Stop the process.

SHELL SCRIPT:

```
echo "Enter the filename:"

read fnm

if [ -e $fnm ]

then

echo $fnm" file exist"

if [ -s $fnm ]

then

echo $fnm" file has size > 0"

else

rm $fnm

echo $fnm" file is deleted which has size = 0"

fi

else
```

```
echo "File does not exist"

fi
```

OUTPUT:

```
1926ka38 @Kgcaslinux lab]$sh program6.sh

Enter the filename:

content.txt

content.txt file exist

content.txt file has size > 0
```

ExNo:07

FINDING THE GREATEST AMONG THE GIVEN NUMBERS USING COMMAND LINE ARGUMENTS

AIM:

To write a shell script to find the greatest among the given number using command line argument.

ALGORITHM:

Step1: Start the process.

Step2: Create a file using vi editor.

Step3: Using echo print the statement.

Step4: Get the file name as input from the user.

Step5: Using the for loop check the condition and print the numbers stored in the array.

Step6: Using if print the statement as greater and smaller numbers.

Step7: Print the smallest numbers and largest numbers.

Step8: The output will be displayed on the screen.

Step9: Read the total number count for an array can be get from user using command line arguments.

Step10: Stop the process.

SHELL SCRIPT:

```
for((i=0;i<$1;i++))

do

echo "Enter $((i+1)) number:"

read nos[$i]

done

echo "Number entered are:"

for((i=0;i<$1;i++))

do
```

```
echo ${nos[$i]}
done
small=${nos[0]}
greater=${nos[0]}
for((i=0;i<$1;i++))
do
if [ ${nos[$i]} -lt $small ];
then
small=${nos[$i]}
elif [ ${nos[$i]} -gt $greater ];
then
greater=${nos[$i]}
fi
done
echo "Smallest number in an array is $small"
echo "Greatest number in an array is $greater"
```

OUTPUT:

```
[1922kc62@Kgcaslinux lab]$sh program8.sh 3
Enter 1number
6
Enter  2number
3
Enter  3number
9
Numbers entered are:
6
3
9
Smallest number in an array is 3
Greatest number in an array is 9
```

ExNo:08

FINDING THE SUM OF INDIVIDUAL DIGITS OF GIVEN NUMBER

AIM:

To write a shell script to find the sum of individual digits.

ALGORITHM:

Step 1: Start the process**.**

Step 2: Create a file using vi editor.

Step 3: Using echo print the statement “Enter the number”.

Step 4: Read the value from user whose summation of individual digits can be found.

Step 5: Declare & initialize the variables sd=0 and sum=0.

Step 6: Find n%10,n/10,sum=sum + sd and store it in sd, n, sum respectively.

Step 5: Repeat the step 6 until n greater than 0 using while loop.

Step 6: Display the output.

Step 7: Stop the process

SHELL SCRIPTS:

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

read n

sd=0

sum=0

while [ $n -gt0 ]

do

sd=$(( $n % 10 ))

n=$(( $n / 10 ))

sum=$(( $sum + $sd ))

done

echo "Sum of all digits is "$sum
```

OUTPUT:

```
[1922kc62@Kgcaslinux lab]$sh program7.sh

Enter a number:1234

Sum of all digits is10
```

ExNo:09

SHELL SCRIPT FOR CECKING THE GIVEN STRING OR NUMBER IS PALINDROME OR NOT

AIM:

To write a shell script for palindrome checking.

ALGORITHM:

Step 1: Start the process.

Step 2: Get the string from the user.

Step 3: Reverse the string using rev<<<string

Step 4: If the reversed string and the given string is same display it as palindrome

Step 5: Else display not a palindrome for this use if else.

Step 6: Stop the process

SHELL SCRIPT:

```
read -p "Enter a string:" string

if [[ $(rev <<< "$string") == "$string" ]];

then

echo "palindrome"

else

echo "not a palindrome"

fi
```

OUTPUT:

Enter a string: 1221

Number is palindrome

ExNo:10

PRINTING THE MULTIPLICATION TABLE USING FOR LOOP

AIM:

To write a shell script to print multiplication tables.

ALGORITHM:

Step 1: Start the process.

Step 2: Get the table number and range from the user.

Step 3: Use for loop to find the table of the given number and range.

Step 4: Display the calculated table in the standard table format.

Step 5: Stop the process

SHELL SCRPIT:

```
echo "Enter the table number"

read n

echo "Enter the range"

read range

echo "Multiplication table for $n upto the range $range"

for((i=1;i<=range;i++))

{

echo " $i X $n = `expr $n \* $i`"

}
```

OUTPUT:

```
Enter the table number

1

Enter the range

4

Multiplication table for 1 upto the range 4
```

1 X 1 =1

2 X 1 = 2

3 X 1 = 3

4 X 1 =4