EXPERIMENT-8

SHELL SCRIPT

Write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, have the program ask the user for the necessary information, such as the file name, new name and so on.

Aim: To write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, have the program ask the user for the necessary information, such as the file name, new name and so on.

Description:

CP Command:

cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. *cp* command require at least two filenames in its arguments.

Syntax:
cp [OPTION] Source Destination
cp [OPTION] Source Directory
cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

MV Command:
mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX. It has two distinct functions:
(i) It renames a file or folder.
(ii) It moves a group of files to a different directory.
Syntax:
mv [Option] source destination

RM Command:
rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.
Syntax:
rm [OPTION]... FILE...

LN Command:

The *ln* command is used to create links between files. Before going into the application of the ln command in detail, please refer the below link for a clear understanding of the hard link and soft link in Linux.

Syntax:

ln [OPTION]... [-T] TARGET LINK_NAME (1st form)

ln [OPTION]... TARGET... DIRECTORY (2nd form)

ln [OPTION]... -t DIRECTORY TARGET... (3rd form)

```
[20A91A05I3@Linux ~]$ vi file.sh
```

20A91A05I3@Linux:~

```
echo 1.copy
echo 2.rename
echo 3.remove
echo 4.linkn
echo 5.exit
echo "enter your choice"
read ch
case $ch in
1) echo "enter the source file"
read s
echo "enter the destination file"
read d
cp $s $d
;;
2) echo "enter old file name"
read of
echo "enter the new file name"
read nf
mv $of $nf
;;
3) echo "enter file name to delete"
read df
rm $df
;;
4) echo "enter file1"
read f1
echo "enter file2"
read f2
ln $f1 $f2
;;
5) exit 0
;;
esac
~
~
```

```
[20A91A05I3@Linux ~]$ sh file.sh
1.copy
2.rename
3.remove
4.linkn
5.exit
enter your choice
2
enter old file name
file2
enter the new file name
sra
[20A91A05I3@Linux ~]$ cat file2
cat: file2: No such file or directory
[20A91A05I3@Linux ~]$ cat sra
hi
how are you
```

```
[20A91A05I3@Linux ~]$ sh file.sh
1.copy
2.rename
3.remove
4.linkn
5.exit
enter your choice
1
enter the source file
file1
enter the destination file
file2
[20A91A05I3@Linux ~]$ cat file1
hi
how are you
[20A91A05I3@Linux ~]$ cat file2
hi
how are you
```

```
[20A91A05I3@Linux ~]$ sh file.sh
1.copy
2.rename
3.remove
4.linkn
5.exit
enter your choice
3
enter file name to delete
sra
[20A91A05I3@Linux ~]$ cat sra
cat: sra: No such file or directory
```

```
[20A91A05I3@Linux ~]$ sh file.sh
1.copy
2.rename
3.remove
4.link
5.exit
enter your choice
4
enter file1
file1
enter file2
sravya
[20A91A05I3@Linux ~]$ ls -l file1
-rw-rw-r-- 2 20A91A05I3 20A91A05I3 15 Nov  9 11:59 file1
[20A91A05I3@Linux ~]$ ls -l sravya
-rw-rw-r-- 2 20A91A05I3 20A91A05I3 15 Nov  9 11:59 sravya
```

```
[20A91A05I3@Linux ~]$ sh file.sh
1.copy
2.rename
3.remove
4.link
5.exit
enter your choice
5
[20A91A05I3@Linux ~]$
```

EXPERIMENT-7

SHELL SCRIPT

a) Write a shell script that accepts two integers as its arguments and computes the value of first number raised to the power of the second number.

Aim: To write a shell script that accepts two integers as its arguments and computes the value of first number raised to the power of the second number.

Description:

Command Arguments and Parameters:

The Unix shell is used to run commands, and it allows users to pass run time arguments to these commands.

These arguments, also known as command line parameters, that allows the users to either control the flow of the command or to specify the input data for the command.

How to work with command line parameters:

While running a command, the user can pass a variable number of parameters in the command line.

Within the command script, the passed parameters are accessible using 'positional parameters'. These range from $0 to $9, where $0 refers to the name of the command itself, and $1 to $9 are the first through to the ninth parameter, depending on how many parameters were actually passed.

The command-line arguments $1, $2, $3, ...$9 are positional parameters, with $0 pointing to the actual command, program, shell script, or function and $1, $2, $3, ...$9 as the arguments to the command**.**

```
echo "File Name: $0"

echo "First Parameter : $1"

echo "Second Parameter : $2"

echo "Quoted Values: $@"
```

echo "Quoted Values: $*"

echo "Total Number of Parameters : $#"

Exit : $echo $?

some additional commands to process these parameters.

1) set : This command can be used to set the values of the positional parameters on the command line.

Example 1 : kalyan@kalyan-Inspiron-N5110:~$ set kalyan

kalyan@kalyan-Inspiron-N5110:~$ echo $0

bash

kalyan@kalyan-Inspiron-N5110:~$ echo $1

kalyan

kalyan@kalyan-Inspiron-N5110:~$ set kalyan ram aditya

kalyan@kalyan-Inspiron-N5110:~$ echo $*

kalyan ram aditya

kalyan@kalyan-Inspiron-N5110:~$ echo $#

3

echo Total Number of Argument Passed: "$#"

echo All Arguments are: "$*"

BC Command:

bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.
Arithmetic operations are the most basic in any kind of programming language. Linux or Unix operating system provides the bc command and expr command for doing arithmetic calculations. You can use these commands in bash or shell script also for evaluating arithmetic expressions.
Syntax:
bc [-hlwsqv] [long-options] [file ...]
Options:
-h, {- -help } : Print the usage and exit

-i, {- -interactive } : Force interactive mode
-l, {- -mathlib } : Define the standard math library
-w, {- -warn } : Give warnings for extensions to POSIX bc
-s, {- -standard } : Process exactly the POSIX bc language
-q, {- -quiet } : Do not print the normal GNU bc welcome
-v, {- -version } : Print the version number and copyright and quit

```
[20A91A05I3@Linux ~]$ vi she.sh
```

20A91A05I3@Linux:~

```
if [ $# -ne 2 ]
then
echo "invalid number of arguments"
exit
fi
pwr=`echo $1^$2|bc`
echo "$1 raised to $2 is $pwr"
~
~
```

```
[20A91A05I3@Linux ~]$ sh she.sh
invalid number of arguments
[20A91A05I3@Linux ~]$ sh she.sh 9 3
9 raised to 3 is 729
```

EXPERIMENT-9

b) Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

Aim: To write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

Description:

CMP Command:

cmp command in Linux/UNIX is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.

When cmp is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found *i.e* the files compared are identical.
cmp displays no message and simply returns the prompt if the the files compared are identical.
Syntax:
cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

RM Command:
rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.
Syntax:
rm [OPTION]... FILE...

if statement:
This block will process if specified condition is true.
Syntax:
```
if [ expression ]
then
  statement
fi
```

```
[20A91A05I3@Linux ~]$ vi sha.sh
```

20A91A05I3@Linux:~

```
echo enter first file
read f1
echo enter second file
read f2
`cmp $f1 $f2>equal`
if [ ! -s equal ]
then
echo files are equal
rm $f2
echo removed $f2
else
echo files are not equal
fi
~
~
~
```

```
[20A91A05I3@Linux ~]$ sh sha.sh
enter first file
file1
enter second file
shs
files are equal
removed shs
[20A91A05I3@Linux ~]$ cat shs
cat: shs: No such file or directory
```

```
[20A91A05I3@Linux ~]$ sh sha.sh
enter first file
file1
enter second file
123
files are not equal
```