Practical -2 Basic file command & basic shell script

- Echo command

Display message on the screen

syntax : **echo** [*OPTION*]... [*STRING*]

-n do not output the trailing newline

-e to use backspace character

\\ backslash

\b backspace

\n new line

Example :

1. echo $HOME
   /home/student
2. echo "this is \n new line"
   this is \n new line
3. student@mcastaff:~$ echo -e "this is \n new line"
   this is
   new line
4. student@mcastaff:~$ echo -e "testing of \bback slash"
   testing ofback slash

- wc command

print newline, word, and byte counts for each file

Syntax : wc [OPTION]... [FILE].

Option :

-w : count number of words

-l : count number of lines

-c : count number of bytes

-L : display maximum length line width

Example

$ wc file1 fiel2

$ wc -l file2

$ wc -w file2

$ wc – L file2

- ls command

it list out the content of directory.

Syntax : ls [options] file/directory name

Options :

-R Recursively list subdirectories encountered.

-a Include files with a name starting with `.' in the listing. Include all

-i Precede the output for the file by the file i-node number.

-r : Reverse the order of the sort.

-t : Sort by the timestamp shown.

-u : Use the time of last access instead of the modification time for sorting

-lu : sorts listing by last access time

-l it show attributes of file like , file type, the number of links to the file, the owner name, the group name, the size of the file (in bytes) , last modified time, file name

-S sort by file size

Example
$ls
$ls – i // display file name with inode number
$ ls – t // display file in sorted order according to modification time(latest modified display first)
$ ls -r
$ ls -R
$ls – lu // display file in sorted order according to most recently accessed file
display first form current directory

$ cat > .t
Testing of hidden file
^c

$ ls -a
$ ls -lS // display file in sorted order according to size of file

- ps command

Ps display the processes owned by the user running command.
Syntax : ps [option]

- Wild card characters (used with l)

*	Any no of characters including none
?	A single character
[ijk]	A single character – either i , j , k
[x-z]	A single character within ascii range [x and z]
[!ijk]	A single character that is no i, j, or k
[!x-z]	A single character that is not within x and z

$ ls f*
$ls file[1-9]
$ cat > first
File content here
^c
$ls f*t // first charcter is f and last is t

$ls [!ft] * // first character is not f or t

$ ls *.txt
$ ls ??? // find the file having only 3 character long
$ ls [ftr]* // first latter is f or t or r in file name

- **chmod command**

Command is used to change the permission of the specified file(s).

There are three types of permissions .
R : Read permission
W : Write permission
X : execute permission

Octal represntation of permission

Binary	Octal	Permission
001	1	Execute only
010	2	Write only
011	3	Write and Execute only
100	4	Read only
101	5	Read and execute
110	6	Read and write
111	7	Read , write & execute

student@mcastaff-VirtualBox:~/Desktop/dir$ ls -l
-rw-rw-r-- 1 student student 75 Jan 9 09:16 hello.c
-rw-rw-r-- 1 student student 38 Jan 9 09:14 new.txt
-rw-rw-r-- 1 student student 49 Jan 9 09:14 two.txt

Permission string brekup into three groups
rw- rw- r- -

First group : permission for owner
Second group : Permission for group
Third group : Permission for other.

student@mcastaff-VirtualBox:~/Desktop/dir$ chmod 444 new.txt

first 4 is for user. Have read only permission
second 4 is for group memebers of same as user , have read only permission
Third 4 is for Others . Have read only permission for new.txt file

Output
student@mcastaff-VirtualBox:~/Desktop/dir$ ls -l
-rw-rw-- 1 student student 75 Jan 9 09:16 hello.c
-r--r--r-- 1 student student 38 Jan 9 09:14 new.txt
-rw-rw-r-- 1 student student 49 Jan 9 09:14 two.txt

e.g 2
student@mcastaff-VirtualBox:~/Desktop/dir$ chmod 777 hello.c
student@mcastaff-VirtualBox:~/Desktop/dir$ ls -l

-rwxrwxrwx 1 student student 75 Jan 9 09:16 hello.c
-r--r--r-- 1 student student 38 Jan 9 09:14 new.txt
-rw-rw-r-- 1 student student 49 Jan 9 09:14 two.txt

- **Shell script basic**

- A shell program runs in interpretive mode.
- It is not compiled to a separate executable file . Each statement is loaded into memory when it is to be executed.
- When a group of command have to be executed regularly, it has should be stored in a file. and file it self run as a shell script.
- It is stored with .sh extention.
- Shell script are executed in a separate child shell process. And sub shell need not be the same as the login shell.
-
- # is used for the comment. What ever characters after # sign is ignored by the shell.
- #! Is followed by the path name of the shell to be used in shell script.
- The first line of shell script is called the interpreter line which specify the type of shell used by script
- **# can not be the first line of the shell script.**

- Create shell script
  $ gedit first.sh
  ```
  #! /bin/bash
  # first.sh
  echo "welcome"
  echo "Today is: `date`    "
  ```

- Run the shell script
  $ sh <scriptname> [arguments]
  e.g $ sh first.sh

e.g 2 two.sh

```
#! /bin/bash
# two.sh
echo "home directory is $HOME"
echo "current working directory is `pwd`"
```

Output
$sh two.sh

```
home directory is /home/student
current working directory is /home/student/script
```

e.g 3 Variable declare and display value

```
#! /bin/bash
echo "Testing of variable"
var=10
echo "value is $var"
```

Output
$sh vartest.sh

```
  Testing of variable
```

value is 10

- Reading value of variable form user

  read <variable name> : is used to read the value form the user and store it into variable name

  e.g
  ```
  #! /bin/bash
  echo "Enter the value"
  read val
  read
  echo $val
  #read v1 v2
  #echo $v1 $v2
  ```

  ```
  Enter the value
  test
  test
  student@mcastaff:~/script$ sh readtest.sh
  Enter the value
  5
  5
  ```

e.g read the file name from user and display the content of the file

```
#!/bin/bash
#displaying  file content of existing fille
echo "enter existing file name "
read fname
echo "content of the file is "
echo "-----------------"
cat $fname
```

Output

```
$sh ftest.sh
Enter existing file name
 File1
 content of the file
 ------------------------
 anc
  qrts
```

- expr for arithmetic operation in shell script

  expr is used for performing arithmetic operation in shell script.

  ```
  #! /bin/bash
  # exprtest.sh
  echo "program for testing  addition operation"
  echo "enter no1"
  read no1
  echo "enter no2"
  read no2

  echo "addition without expr"
  no3=$no1 + $no2
  ```

```
echo $no3
echo "addition with expr"
no3=`expr $no1 + $no2`

echo $no3
```

Output :

```
student@mcastaff:~/script$ sh exprtest.sh
program for testing  addition operation
enter no1
3
enter no2
4
addition without expr
exprtest.sh: 10: exprtest.sh: +: not found   // error

addition with expr
7
```

- Command line argument
- Using the command line shell script can run noninteractively
- When arguments are specified with a shell script they are assigned to certain special variable.
- The first argument is read by shell into **$1** parameter
- Second argument is read into **$2** parameter like wise.. All the variables is read

Parameters	**Significance**
$1, $2	Positional parameters
$#	Number of arguments in command line
$0	Name of executed command
$*	Complete set of positional parameter
“$@”	Each quoted string as a separate arguments

Eg. Script for testing the command line arguments.

```
#! /bin/bash
# testing command line arguments
echo "program name is  : $0"
echo "The number of arguments is $# "
echo "The arguments are $* "
echo "The first argument is $1 "
echo "the second argument is $2 “
```

Output

```
student@mcastaff:~$ sh cmdtest1.sh "hello" "second" 23 45 "new"
cmdtest1.sh: 1: cmdtest1.sh: Script: not found
program name is  : cmdtest1.sh
The number of arguments is 5
The arguments are hello second 23 45 new
The first argument is hello
the second argument is second
```

Testing of $@ is hello second 23 45 new

e.g 2 head and tail command applied to file passed as command line argument

```
#! /bin/bash
# head command  displays top lines of the file
# tail command displays bottom lines of the file.

echo "the file name is $1"
echo "the first $2 line of $1 are "
echo “----------"
head -$2 $1
echo "the last $3 line of $1 are "
tail -$2 $1
```

Output

```
the file name is file2
the first 2 line of file2 are
-------
enter the the data



the last  line of file2 are
new line
new line is appended
```

Exercise

1. List out all file start and ends with digit.
2. count the number of lines of all the files available in newdir.
3. list out all the file having txt in file name
4. List out all the file in reverse order alphabetically sort.
5. List out all the file whose name does not start with r or s.
6 . List out all the file whose name ends with t or p.
7. change the permission : give read and write permission to owner, read only permission to group and other.
8. Write a shell script which present working directory and current active user name.(print output with proper message)
9. Write a shell script which pass the file name as argument and display its content , display number of lines and number of words of a file.
10. Write a shell script which pass two argument and perform all arithmetic operation on two number. (use * for multiplication)
11. Write a shell script which take filename as a input form user and move to other directory.
12. Write a shell script which take month and year as input from user and display calendar of given year’s month.
13 . Write a shell script which pass two arguments and rename the file passed in first argument, with second argument name.
14. write a shell script which read a directory name of user and display all the information(attribute) of files inside given directory.