VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY
(Beside VSEZ,Duvvada,Gajuwaka,VSKP-530049.A.P.)
DEPARTMENT OF IT
FREE OPEN SOURCE SOFTWARE (FOSS) LAB
II Year B.Tech II Sem

T	P	C
0	3	2

OBJECTIVES:

To enable students learn and know various unix utilities.
To make students get hands on experience on shell scripting.

Programs:

1.

Session-1:

a)Log into the system
b)Use vi editor to create a file called myfile.txt which contains some text.
c)correct typing errors during creation. d)Save the file
e)logout of the system

Session-2:

a)Log into the system
b)open the file created in session 1
c)Add some text
d)Change some text
e)Delete some text
f)Save the Changes
g)Logout of the system

2.

a)Log into the system
b)Use the cat command to create a file containing the following data. Call it mytable use tabs to separate the fields.

1425	Ravi	15.65
4320	Ramu	26.27
6830	Sita	36.15
1450	Raju	21.86

c)Use the cat command to display the file, mytable.
d)Use the vi command to correct any errors in the file, mytable.
e)Use the sort command to sort the file mytable according to the first field. Call the sorted file my table (same name) f)Print the file mytable
g)Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it my table (same name)
h)Print the new file, mytable i)Logout of the system.

3.

1) a)Login to the system
b)Use the appropriate command to determine your login shell
c)Use the /etc/passwd file to verify the result of step b.
d)Use the who command and redirect the result to a file called myfile1. Use the more command to see the contents of myfile1.
e)Use the date and who commands in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2. Use the more command to check the contents of myfile2.

2) a)Write a sed command that deletes the first character in each line in a file.
b)Write a sed command that deletes the character before the last character in each line in a file.
c)Write a sed command that swaps the first and second words in each line in a file.

4. a)Pipe your /etc/passwd file to awk, and print out the home directory of each user.
b)Develop an interactive grep script that asks for a word and a file name and then tells how many lines contain that word. c)Repeat
d)Part using awk

5. a)Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else. b)Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.
c)Write a shell script that determines the period for which a specified user is working on the system.

6. a)Write a shell script that accepts a file name starting and ending line numbers as arguments and displays all the lines between the given line numbers.
b)Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

7. a)Write a shell script that computes the gross salary of a employee according to the following rules:
i)If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic.
ii)If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic
The basic salary is entered interactively through the key board. b)Write a shell script that accepts two integers as its arguments and computers the value of first number raised to the power of the second number.

8. a)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.
b)Write shell script that takes a login name as command – line argument and reports when that person logs in
c)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.

9. a)Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.
b)Develop an interactive script that ask for a word and a file name and then tells how many times that word occurred in the file.
c)Write a shell script to perform the following string operations:
i)To extract a sub-string from a given string.
ii)To find the length of a given string.

10. Write a C program that takes one or more file or directory names as command line input and reports the following information on the file:
i)File type
ii)Number of links
iii)Read, write and execute permissions
iv)Time of last access (Note : Use stat/fstat system calls)

11. Write C programs that simulate the following unix commands:
a)mv b)cp (Use system calls)

12. Write a C program that simulates ls Command (Use system calls / directory API)

13. Do the following Shell programs also
1) Write a shell script to check whether a particular user has logged in or not. If he has logged in, also check whether he has eligibility to receive a message or not
2) Write a shell script to accept the name of the file from standard input and perform the following tests on it
a) File executable b) File readable c) File writable
d) Both readable & writable
3) Write a shell script which will display the username and terminal name who login recently in to the unix system
4) Write a shell script to find no. of files in a directory
5) Write a shell script to check whether a given number is perfect or not
6) Write a menu driven shell script to copy, edit, rename and delete a file
7) Write a shell script for concatenation of two strings
3) Write a shell script which will display Fibonacci series up to a given number of argument
9) Write a shell script to accept student number, name, marks in 5 subjects. Find total, average and grade. Display the result of student and store in a file called stu.dat
Rules: avg>=80 then grade A Avg<80&&Avg>=70 then grade B
Avg<70&&Avg>=60 then grade C Avg<60&&Avg>=50 then grade
D Avg<50&&Avg>=40 then grade E Else grade F
10) Write a shell script to accept empno,empname,basic. Find DA,HRA,TA,PF using following rules. Display empno, empname, basic, DA,HRA,PF,TA,GROSS SAL and NETSAL. Also store all details in a file called emp.dat
Rules: HRA is 18% of basic if basic > 5000 otherwise 550

DA is 35% of basic PF is 13% of basic IT is 14% of basic TA is 10% of basic

11) Write a shell script to demonstrate break and continue statements
12) Write a shell script to satisfy the following menu options
    a. Display current directory path
    b. b. Display todays date
    c. Display users who are connected to the unix system
    d. d. Quit
13) Write a shell script to delete all files whose size is zero bytes from current directory.
14) Write a shell script to display string palindrome from given arguments
15) Write a shell script which will display Armstrong numbers from given arguments
16) Write a shell script to display reverse numbers from given argument list
17) Write a shell script to display factorial value from given argument list
18) Write a shell script which will find maximum file size in the given argument list
19) Write a shell script which will greet you "Good Morning", "Good Afternoon", "Good Evening' and "Good Night" according to current time
20) Write a shell script to sort the elements in a array using bubble sort technique
21) Write a shell script to find largest element in a array
22) Write an awk program to print sum, avg of students marks list.
23) Write an awk program to display students pass/fail report
24) Write an awk program to count the no. of vowels in a given file
25) Write an awk program which will find maximum word and its length in the given input File
26) Write a shell script to generate the mathematical tables.
27) Write a shell script to sort elements of given array by using selection sort.
28) Write a shell script to search given number using binary search.
29) Write a shell script to find number of vowels, consonants, numbers, white spaces and special characters in a given string.
30) Write a shell script to lock the terminal.

FREE AND OPEN SOURCE SOFTWARE:

Free & Open Source Software describes software that is licensed with fewer restrictions than proprietary licensing models, such as "per copy", "per use" object code only licenses.

The term "free software" often refers to software that is licensed under the General Public License ("the GPL").

"Free" does not refer to cost, as the GPL does not preclude charging for distribution of licensed software, but rather it refers to the lack of constraints on using the software.

The term "open source" often refers to free software as well as software licensed under other licenses generally considered to be open source licenses. Open source licenses might include provisions regarding limits on constraints, attribution requirements, no-warranty notices and other important provisions.

Open source software often has fewer constraints on intermediate parties, but possibly more constraints on downstream parties

EXAMPLES OF FREE AND OPEN SOURCE SOFTWARE:

Linux is perhaps the most widely known example of open source software.

The Apple Macintosh's underlying operating system is open source software.

Other examples include the Apache Web Server programs used by many of the web servers running today.

The GNU operating system, including its many programming tools, development environments and programs, is also free software.

A BRIEF HISTORY OF UNIX:

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities. The following is a simplified history of how UNIX has developed

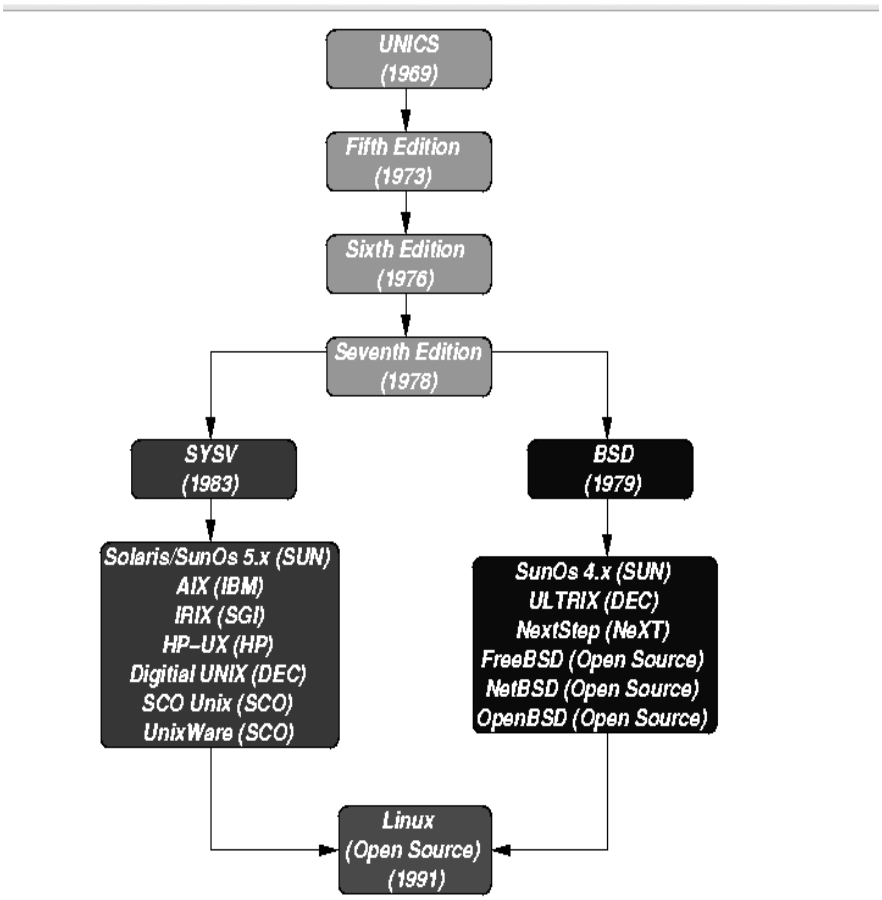


Fig: A Simplified UNIX Tree

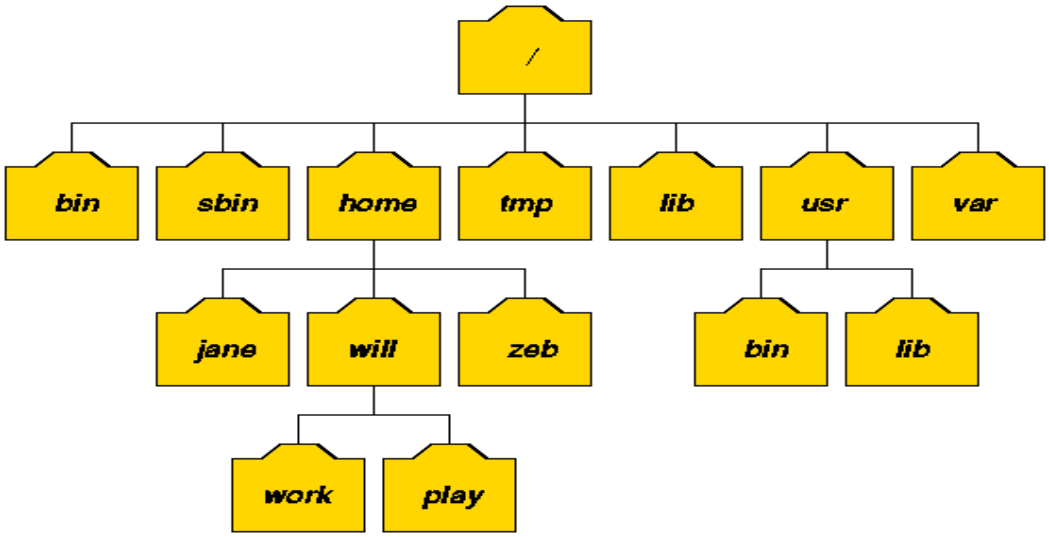
LINUX:

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution.

Linux is a free open source UNIX OS for PCs that was originally developed in 1991 by Linus Torvalds, a Finnish undergraduate student.

The open source nature of Linux means that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies. This approach has been very successful and what started as one person's project has now turned into a collaboration of hundreds of volunteer developers from around the globe.
The open source approach has not just successfully been applied to kernel code, but also to application programs for Linux.

UNIX DIRECTORY STRUCTURE



Directory	Typical Contents
/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level system utilities
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes	User home directories containing personal file space for each user. Each directory is named after the login of the user.
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel. Includes a sub-directory for each active program (or process).

FILE HANDLING COMMANDS:

1. pwd (print [current] working directory)
   pwd displays the full absolute path to the your current location in the filesystem. So
   $ pwd ↵
   /usr/bin
   implies that /usr/bin is the current working directory.

2. ls (list directory)
   ls lists the contents of a directory. If no target directory is given, then the contents of the current working directory are displayed. So, if the current working directory is /,
   $ ls ↵
   bin dev home mnt share usr var
   boot etc lib proc sbin tmp vol

3. man
   $ man ls ↵
   man is the online UNIX user manual,

4. mkdir (make directory)
   $ mkdir *directory*
   creates a subdirectory called *directory*in the current working directory. You can only create subdirectories in a directory if you have write permission on that directory.

5. cd (change [current working] directory)
   $ cd *path*
   changes your current working directory to *path* (which can be an absolute or a relative path). One of the most common relative paths to use is '..' (i.e. the parent directory of the current directory).

6. rmdir (remove directory)
   $ rmdir *directory*
   removes the subdirectory *directory* from the current working directory. You can only remove subdirectories if they are completely empty (i.e. of all entries besides the '.' and '..' directories).

7. cp (copy)
   cp is used to make copies of files or entire directories. To copy files, the syntax is:
   $ cp *source-file(s) destination*

8. mv (move/rename)
   mv is used to rename files/directories and/or move them from one directory into another. Exactly one source and one destination must be specified:
   $ mv *source destination*

9. rm (remove/delete)
   $ rm *target-file(s)*
   removes the specified files.

OTHER UNIX COMMANDS:

UNIX Command	Description
Sh	Invokes Korn shell.
date -	shows the current date and time
Cal	shows a calendar of the current month. Use e.g., 'cal 10 1995' to get that for October 95, or 'cal 1995' to get the whole year.
Whoami	returns your username. Sounds useless, but isn't. You may need to find out who it is who forgot to log out somewhere, and make sure *you* have logged out.
Who	tells you who's logged on, and where they're coming from. Useful if you're looking for someone who's actually physically in the same building as you, or in some other particular location.
Cat	Concatenates and displays file.
Dirname	Delivers all but the last level of the path in string. See *basename* .
Find	Recursively searches directory hierarchy looking for files that match a specified Boolean expression.
Grep	Searches files for a pattern and prints all line containing that pattern.
Ls	Lists contents of a directory.
Mkdir	Creates named directory with read, write, and execute permission for every type of user.
More	A filter that displays the contents of a text file on the terminal, one screen at a time.
Sed	Stream editor. Copies named file name to standard output, edited according to a script of commands.
Sort	Sorts lines of all named files together and writes result to standard output.
Wc	Displays a count of lines, words, or characters in a file.

Vi	Screen-oriented visual display editor based on ex.

INTRODUCTION TO VI:

vi is a display-oriented text editor based on an underlying line editor called ex. Although beginners usually find vi somewhat awkward to use, it is useful to learn because it is universally available (being supplied with all UNIX systems). It also uses standard alphanumeric keys for commands, so it can be used on almost any terminal or workstation without having to worry about unusual keyboard mappings. System administrators like users to use vi because it uses very few system resources.

To start vi, enter:

$ vi *filename* ↵

where *filename* is the name of the file. If the file doesn't exist, vi will create it

.

Starting and Quitting vi

vi*file-name*	Edits *file-name* (this creates a new file or edits an existing one)
vi -r*file-name*	Recovers a file after a system crash and edit it
:q	Quits vi if no changes have been made
:q!	Quits vi without saving changes
:wq	Writes (save changes) and quit vi

Input Mode Commands

A	Insert text after the cursor
A	Insert text at end of the current line
i	Insert text before the cursor
I	Insert text before the current line
o	Open a line in the text below the cursor
O	Open a line in the text above the cursor

Changing Text

r	Replace the current character with the next character typed; return to Command mode.
R	Replace text beginning with the current character, until ESC invoked.
Cc	Changes the entire current line to the new text entered.
Cw	Changes the current word, beginning at the cursor position, to the new text entered.
s	Substitutes the character at the cursor position with the new text entered.

S Substitutes the entire current line with the new text entered.

Deleting Text

D Delete from cursor to the end of the line

X Delete the current character

Dd Delete the current line

Search Commands:

/*pattern*	Moves forward to the first character in the next occurrence of the character string *pattern* .
/	Repeats the previous forward search.
?*pattern*	Moves backward to the first character in the next occurrence of the character string *pattern* .
?	Repeats the previous backward search.

SED:

Sed Command in Linux:

Sed is a Stream Editor used for modifying the files in unix (or linux).

A stream editor is used to perform basic text transformations on an input stream. **sed** maintains two data buffers: the active *pattern space*, and the auxiliary *hold space*. Both are initially empty.**sed** operates by performing the following cycle on each line of input: first, **sed** reads one line from the input stream, removes any trailing newline, and places it in the pattern space. Then commands are executed; each command can have an address associated to it: addresses are a kind of condition code, and a command is only executed if the condition is verified before the command is to be executed.

The following table lists four special characters that are very useful in regular expressions.

Character	Description
^	Matches the beginning of lines.

$	Matches the end of lines.
.	Matches any single character.
*	Matches zero or more occurrences of the previous character
[chars]	Matches any one of the characters given in chars, where chars is a sequence of characters. You can use the - character to indicate a range of characters.

Following table shows some frequently used sets of characters:

Set	Description
[a-z]	Matches a single lowercase letter
[A-Z]	Matches a single uppercase letter
[a-zA-Z]	Matches a single letter
[0-9]	Matches a single number
[a-zA-Z0-9]	Matches a single letter or number

Matching Characters:

Expression	Description
/a.c/	Matches lines that contain strings such as a+c, a-c, abc, match, and a3c, whereas the pattern
/a*c/	Matches the same strings along with strings such as ace, yacc, and arctic.
/[tT]he/	Matches the string The and the:
/^.*$/	Matches an entire line whatever it is.
/ */	Matches one or more spaces

SHELL:

A shell is a program which reads and executes commands for the user. Shells also usually provide features such job control, input and output redirection and a command language for writing *shell scripts*.

SHELL SCRIPT:

A shell script is simply an ordinary text file containing a series of commands in a shell command language

DIFFERENT TYPENT TYPES OF SHELLS:

There are many different shells available on UNIX systems
(e.g. sh, bash, csh, ksh, tcsh etc.), and they each support a different command language.

Program1:

Session 1:

a)Log into the system
b)Use vi editor to create a file called myfile.txt which contains some text.
c)Correct typing errors during creation.
d)Save the file
g)Logout of the system

Solution:

a) Log into the system

Username:user1
Password:

Steps:

1.Create directory by using following command
MKDIR IT
2. To verify whether it is created or not
DIRS (or) ls –a
3. To switch your own directory from root directory type the following command
cd IT
4. Now it looks like below prompt
/root/IT/

b) Use vi editor to create a file called myfile.txt which contains some text.

$vi myfile.txt

WELCOME TO FOSS LAB
FOSS STANDS FOR FREEE OPEN SOURCE SOFTWARE

<esc>:wq

c) Correct typing errors during creation.

Original file:

WELCOME TO FOSS LAB
FOSS STANDS FOR FREEE OPEN SOURCE SOFTWARE

Modifications:

vi myfile.txt

Press i:Use down arrow to move to third line and correct the word freee to free use delete key

WELCOME TO FOSS LAB
FOSS STANDS FOR FREE OPEN SOURCE SOFTWARE

d) Save the file

Press esc :w – save all changes made so far.
Press esc :we -Save all changes and quit.
Press esc :q! - Quit without saving changes

e) Logout of the system

$ exit

Session 2:

a) Log into the system
b) Open the file created in session 1
c) Add some text
d) Change some text
e) Delete some text
f) Save the Changes
g) Logout of the system

Solution:

a) Log into the system

Login :user1
password:

b) Open the file created in session 1

In the above session we created a file in vi editor To open the file created in Session1 we have to type

$vi myfile

It displays the file as follows:

WELCOME TO FOSS LAB
FOSS STANDS FOR FREE OPEN SOURCE SOFTWARE

c) Add some text

To add some text to the file which is already created first we have to open that file then add text.

vi myfile. txt

WELCOME TO FOSS LAB
FOSS STANDS FOR FREE OPEN SOURCE SOFTWARE

Press <i> to enter into insert mode. Press <A> to append after the current character. Enter the following lies.

WE WORK IN UNIX ENVIRONMENT
WE HAVE FOSS LAB EVERY WEEK

Press<esc>:wq to quit vi editor.

d) Change some text

To change the some text in myfile.text move the cursor where we want to change the text.

vi myfile.text

WELCOME TO FOSS LAB
FOSS STANDS FOR FREE OPEN SOURCE SOFTWARE
WE WORK IN UNIX ENVIRONMENT
WE HAVE FOSS LAB EVERY WEEK

After that replace the text under cursor with other text .Then open the file again to see the changes made.

WELCOME TO FOSS LAB
FOSS STANDS FOR FREE OPEN SOURCE SOFTWARE
WE WORK IN UNIX ENVIRONMENT
WE HAVE FOSS LAB EVERY WEEK

e) Delete some text :

To delete text in myfile. txt we first move the cursor to end of that line and then press delete then the line is erased to do this first we open the file. Do the corrections by using the options like d, dd, 2d …. And close the file. Finally, display the file to see the changes.

vi myfile.text

f) Save the changes:

To save the changes made in file myfile.text we press

esc:WQ

g) Logout of the System:

$ exit

Program 2:

a) Log into the system
b) Use the cat command to create a file containing the following data. Call it mytable use tabs to separate the fields.

1425	**Ravi**	**15.65**
4320	**Ramu**	**26.27**
6830	**Sita**	**36.15**
1450	**Raju**	**21.86**

c) Use the cat command to display the file, mytable.
d) Use the vi command to correct any errors in the file, mytable.
e) Use the sort command to sort the file mytable according to the first field. Call the sorted file mytable(same name)
f) Print the file mytable
g) Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it mytable
h) Print the new file, mytable
i) Logout of the system.

Solution:

a) Log into the system

Username:user1
Password:

b) Use the cat command to create a file containing the following data. Call it mytable use tabs to separate the fields.

1425	Ravi	15.65
4320	Ramu	26.27
6830	Sita	36.15
1450	Raju	21.86

$ cat>mytable

1425	Raavi	15.65
4320	Raamu	26.27
6830	Sitaa	36.15
1450	Raaju	21.86

c) Use the cat command to display the file, mytable.

$ catmytable

1425	Raavi	15.65
4320	Raamu	26.27
6830	Sitaa	36.15

1450 Raaju 21.86

d) Use the vi command to correct any errors in the file, mytable.

Step 1: open file.

vi mytable

Step 2: Press i

Press Right arrow to move to the word 'Raavi'

Press 'r' to replace current character.

Repeat the process to make all the corrections.

Step 3: Type 'vi mytable' to notice the changes made.

1425 Raavi 15.65
4320 Raamu 26.27
6830 Sitaa 36.15
1450 Raaju 21.86

Step 4: Press<esc> :wq to quit vi editor

e) Use the sort command to sort the file mytable according to the first field.Call the sorted file mytable(same name)

$ sort mytable

1425 Ravi 15.65
1450 Raju 21.86
4322 Ramu 26.27
6830 Sita 36.15

f) Print the file mytable

$ lp mytable

g) Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it mytable

$ catmytable

1425 Ravi 15.65
4320 Ramu 26.27
6830 Sita 36.15
1450 Raju 21.86

$ cut -f 1 mytable>field1; cut -f 2 mytable>field2; cut -f 3 mytable>field3
$ paste field1 field3 field2 >mytable

$ cat mytable

```
1425   15.65   Ravi
4320   26.27   Ramu
6830   36.15   Sita
1450   21.86   Raju
```

h) Print the new file, mytable

$lp mytable

i) Logout of the system.

$exit

Program 3:

1)

a) Login to the system
b) Use the appropriate command to determine your login shell
c) Use the /etc/passwd file to verify the result of step b.
d) Use the who command and redirect the result to a file called myfile1.
Use the more commands to see the contents of myfile1.
e) Use the date and who command in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2..Use more command to check the contents of myfle2.

Solution:

a) Login to the system

Username:user1
Password:

b) Use the appropriate command to determine your login shell

```
user1@user1:~/cseb$ echo $SHELL

/bin/bash
```

c) Use the /etc/passwd file to verify the result of step b.

```
user1@user1:~/cseb$cd /
user1@user1:~/cseb$ cd etc
user1@user1:~/cseb$ ls
```

d) Use the who command and redirect the result to a file called myfile1.

```
user1@user1:~/cseb$ who >myfile3
user1@user1:~/cseb$ cat myfile3
user1    :0          2015-04-14 08:07 (:0)
user1pts/0        2015-04-14 09:30 (:0)
```

Use the more commands to see the contents of myfile1.

```
user1@user1 :~/cseb$ more myfile3
user1    :0          2015-04-14 08:07 (:0)
user1pts/0        2015-04-14 09:30 (:0)
```

e) Use the date and who command in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2..Use more command to check the contents of myfle2.

```
user1@user1 :~/cseb$ date;who>myfile2

$ vi myfile2
Tue Apr 14 10:44:02 IST 2015
```

2)

a)Write a sed command that deletes the first character in each line in a file.

b)Write a sed that deletes the character before the last character in each line in a file.

c)Write a sed command that swaps the first and second words in each line in a file.

Solution:

a) Write a sed command that deletes the first character in each line in a file.

```
user1@user1 :~$ cat stu
Alice   21    43    78    84    77    => Pass
Mary    23    21    56    58    45    => Pass
Tom     21    22    38    37          => Fail
Dick    25    37    87    97    95    => Pass
Hay     24    15    30    47          => Fail

user1@user1 :~$ sed 's/^.//' stu

lice    21    43    78    84    77    => Pass
ary     23    21    56    58    45    => Pass
om      21    22    38    37          => Fail
ick     25    37    87    97    95    => Pass
ay      24    15    30    47          => Fail
```

b) Write a sed command that deletes the character before the last character in each line in a file.

```
user1@user1 :~$ cat mytable |sed 's/\(.*\)./\1/'

1425    Ravi    15.6
4320    Ramu    26.2
```

6830 Sita36.1
1450 Raju 21.8

c) Write a sed command that swaps the first and second words in each line in a file.

sed -e's/\([^]\+\) *\([^]\+\)/\2 \1/' mytable

ravi 1425 15.65
ramu 4320 26.27
sita 6830 36.15
raju 1450 21.86

Program 4:

a) Pipe your /etc/passwd file to awk, and print out the home directory of each user.

```
user1@user1:~$ cat /etc/passwd | awk -f home.awk
syslog
usbmux
saned
chand
```

b) Develop an interactive grep script that asks for a word and a file name and then tells how many lines contain the word.

```
echo  "Enter the pattern to be searched: "
read  pattern
echo  "Enter the file to be used: "
read  filename
echo  "Searching for $pattern from file $filename"
echo  "The selected records are: "
grep  "$pattern"  $filename
echo  "The no.of lines contains the word( $pattern ) :"
grep  -c  "$pattern"  $filename
```

OUTPUT:

```
user1@user1:~$ sh grep.sh
Enter the pattern to be searched:
echo
Enter the file to be used:
week5a.sh
Searching for echo from file week5a.sh
The selected records are:
echo  "Enter the file name: "
echo $file "---> It is a ORDINARY FILE."
echo  $file "---> It is a DIRCTORY."
echo  $file "---> It is something else."
The no.of lines containing the word( echo ) : 4
```

c)Repeat

d)Part using awk

Program5:

a) Write a shell script that takes a command -line argument and reports on whether it is directory, a file, or something else.

```
echo  "Enter the file name: "
read file
if [  -f  $file ]
then
echo $file "---> It is a ORDINARY FILE."
elif [  -d  $file  ]
then
echo  $file "---> It is a DIRCTORY."
else
echo  $file "---> It is something else."
fi
```

OUTPUT:

```
user1@user1:~$ sh week5a.sh

Enter the file name:
home.awk
home.awk ---> It is a ORDINARY FILE.

user1@user1:~$ mkdir cseb

user1@user1:~$ sh week5a.sh

Enter the file name:
cseb
cseb ---> It is a DIRCTORY.

user1@user1:~$ sh week5a.sh

Enter the file name:
home
home ---> It is something else.
```

5.

c) Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.

```
for file in *
do
if [ -f $file ]
then
echo $file | tr '[a-z]' '[A-Z]'
fi
done
```

OUTPUT:

```
1TONARM.SH
XDG_VTNR=7
A.OUT
ARM.SH
ARM1.SH
ARMARG.SH
BC.SH
BNARY.SH
BINAY.SH
BLAH
BUBBLE.SH
COUNT.AWK
COUNT1.AWK
DEMO.SH
EX
FOOP.SH
FTST.SH
FZ.SH
GREP.SH
HOME.AWK
```

5.

d) Write a shell script that determines the period for which a specified user is working on the system.

```
echo  "Enter the USER NAME : "
read  user
last  $user
```

OUTPUT

```
user1@user1:~$ sh log.sh

Enter the USER NAME :
chand
chand    pts/0       :0           Thu Apr  2 21:22   still logged in
chand    pts/14      :0           Thu Apr  2 20:57   still logged in
chand    :0          :0           Thu Apr  2 20:56   still logged in
chand    pts/1       :0           Thu Apr  2 19:24 - 20:01  (00:37)
chand    :0          :0           Thu Apr  2 19:16 - down   (00:45)
chand    :0          :0           Thu Apr  2 14:29 - down   (00:46)
chand    pts/7       :0           Thu Apr  2 12:12 - 14:19  (02:06)
wtmp begins Thu Apr  2 12:12:45 2015
```

Program 6:

a) Write a shell script that accepts a file name starting and ending line numbers as arguments and displays all the lines between the given line number.

```
echo "Enter file name"
read fname
echo "Enter starting line number"
read s
echo "Enter ending line number"
read n
sed -n $s,$n\p $fname | cat >newfile
cat newfile
```

OUTPUT:

```
Enter starting line number
3
Enter ending line number
18
                1. List of files.
                2. Copying files.
                3. Removing files.
                4. Renaming files.
                5. Linking files.
                6. Hard Link"
echo  "enter your choice "
read  ch
case "$ch" in
 1 ) echo "The list of file names."
          ls  -l ;;
 2 ) echo "Enter the old filename."
      read ofile
      echo "Enter the new file name."
      read nfile
 cp $ofile $nfile  &&  echo "Copied sucessfully."  |
```

6 b) Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

```
echo  "Enter the word to search for all lines :"
read  word
echo  "the file name are $*   ."
for i  in  $*
do
echo  "The name of the file :"  $i
grep  -v   $word   $i
done
```

OUTPUT:

```
user1@user1:~$ sh deline.sh
Enter the word to search for all lines :
e
the file name are    .
user1@user1:~$ sh deline.sh range.sh
Enter the word to search for all lines :
echo
the file name are range.sh    .
The name of the file : range.sh
        read  file
        if [  -f  $file ]
        then
        read  snum
        read  enum
if  [ $snum  -lt  $enum ]
        then
        sed  -n  '  ' $snum',' $enum' 'p'  '  $file
        else
        fi
        else
        fi
```

Program 7:

a) Write a shell script that computes the gross salary of a employee according to the following rules:

i)If basic salary is < 1500 then HRA=10% of the basic and DA=90% of the basic

ii)IF basic salary is >= 1500 then HRA=Rs500 and DA=98% of the basic.

The basic salary is entered interactively through the keyboard

```
echo  "enter the basic salary:"
read  bsal
if [  $bsal  -lt  1500 ]
then
gsal=$((bsal+((bsal/100)*10)+(bsal/100)*90))
echo  "The gross salary : $gsal"
fi
if [ $bsal  -ge  1500 ]
then
gsal=$(((bsal+500)+(bsal/100)*98))
echo  "the gross salary : $gsal"
fi
```

OUTPUT:

```
user1@user1:~$ sh gsal.sh
enter the basic salary:
100
The gross salary : 200
user1@user1:~$ sh gsal.sh
enter the basic salary:
5000
the gross salary : 10400
user1@user1:~$ sh gsal.sh
enter the basic salary:
10000
the gross salary : 20300
user1@user1:~$ sh gsal.sh
enter the basic salary:
1200
The gross salary : 2400
user1@user1:~$ sh gsal.sh
enter the basic salary:
1500
```

the gross salary : 3470

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

```
echo  "Enter the integer value :"
    read int1
    echo  "Enter the power of that integer:"
    read  int2
    pv=$int1
    i=1
    while [ $i -lt $int2 ]
    do
    pv=`expr  $pv  \*  $int1`
    i=`expr  $i  +  1 `
    done
    echo  "The value of first number to the power of the second number :"
    echo  "$pv"
```

OUTPUT:

```
user1@user1:~$ sh power.sh
Enter the integer value :
5
Enter the power of that integer:
2
The value of first number to the power of the second number :
25
user1@user1:~$ sh power.sh
Enter the integer value :
7
Enter the power of that integer:
3
The value of first number to the power of the second number :
343
user1@user1:~$ sh power.sh
Enter the integer value :
19
Enter the power of that integer:
5
The value of first number to the power of the second number :
2476099
```

Program8:

a) Write a 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.

```
echo  "*******MENU*********"
        echo  "
              1. List of files.
              2. Copying files.
              3. Removing files.
              4. Renaming files.
              5. Linking files."
        Echo  "enter your choice "
        read  ch
        case "$ch" in
1 ) echo "The list of file names."
              ls –l
2 ) echo "Enter the old filename."
              read ofile
              echo "Enter the new file name."
              read nfile
                cp $ofile $nfile  &&  echo "Copied sucessfully."  ||  echo
                   "Copied is not possible." ;;
3 ) echo "Enter the file name to remove."
              read rfile
              rm -f $rfile && echo "Successfully removed." ;;
4 ) echo "Enter the old file name."
              read ofile
              echo "Enter the new file name."
              read nfile
                 mv $ofile $nfile   &&   echo "The file $ofile name renamed
                   to $nfile." ||   echo "You cann't Rename the file. “;;
5 )   echo "Enter the original filename."
              read ofile
              echo "Enter the new filename to link a file."
              read  lfile
              ln $ofile $lfile  &&  echo "Creat the linking file
                   Sccessfully."  ||  echo "You cann't Linking the file.";;
6)  echo "Enter the original filename."
              read ofile
              echo "Enter the new filename to link a file."
```

```
            read  lfile
            ln -s $ofile $lfile  &&  echo "Creat the linking file
                Sccessfully."  ||  echo "You cann't Linking the file.";;
* )    echo  "Invalid option."
            Echo  " Enter correct choice."
           esac
```

OUTPUT:

```
user1@user1:~$ sh week8a.sh
*******MENU*********
                    1. List of files.
                    2. Copying files.
                    3. Removing files.
                    4. Renaming files.
                    5. Linking files.
                    6. Hard Link

enter your choice
1
The list of file names.
total 264
-rw-rw-r-- 1 chand chand  236 Mar 30 12:13 1tonarm.sh
-rwxrwxr-x 1 chand chand 8728 Apr  2 10:30 a.out
-rw-rw-r-- 1 chand chand  560 Mar 30 11:57 arm1.sh
-rwxrwxrwx 1 chand chand  497 Mar 30 12:05 armarg.sh
-rw-rw-r-- 1 chand chand  251 Mar 31 19:31 arm.sh
-rwxrwxrwx 1 chand chand  193 Apr  2 13:43 bc.sh
.............

enter your choice
2
Enter the old filename.
studentmarks
Enter the new file name.
stumark
Copied sucessfully.

enter your choice
3
Enter the file name to remove.
stumark
Successfully removed.
```

```
enter your choice
4
Enter the old file name.
home.awk
Enter the new file name.
week4a.awk
The file home.awk name renamed to week4a.awk.

enter your choice
5
Enter the original filename.
week4a.awk
Enter the new filename to link a file.
home.awk
Creat the linking file
                    Sccessfully.
user1@user1:~$ ls -l stu
lrwxrwxrwx 1 chand chand 12 Apr  2 22:02 stu -> studentmarks
```

b) Write a shell script that makes a login name as command-line argument and reports when the person logs in

```
echo  "Enter the USER NAME : "
read  user
last  $user
```

OUTPUT:

```
sh login.sh
Enter the USER NAME :
chand
chand    pts/14      :0          Thu Apr  2 21:56 - 22:13  (00:16)
chand    pts/0       :0          Thu Apr  2 21:22   still logged in
chand    pts/14      :0          Thu Apr  2 20:57 - 21:55  (00:58)
chand    :0          :0          Thu Apr  2 20:56   still logged in
chand    pts/1       :0          Thu Apr  2 19:24 - 20:01  (00:37)
chand    :0          :0          Thu Apr  2 19:16 - down   (00:45)
chand    :0          :0          Thu Apr  2 14:29 - down   (00:46)
chand    pts/7       :0          Thu Apr  2 12:12 - 14:19  (02:06)
wtmp begins Thu Apr  2 12:12:45 2015
```

8.

c) 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.

```
echo "Enter first file name"
read f1
echo "Enter second file name"
read f2
cmp $f1 $f2 && rm $f2
if [ -e $f1 ]
then
if [ ! -e $f2 ]
then
echo "The two files are same"
echo "The second file deleted successfully"
else
echo "The two file contents are different"
echo "You can't remove second file"
fi
else
echo "You should enter existing file"
fi
```

OUTPUT:

```
Enter first file name
st
Enter second file name

stu
cmp: EOF on st
The two file contents are different
You can't remove second file
user1@user1:~$ sh week8c.sh
Enter first file name
week8c.sh
Enter second file name
week8cn.sh
The two files are same
The second file deleted successfully
```

Program 9:

a) Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.

```
echo "The list of File Names in the curent directory."
echo "Which have Read,Write and Execute permisions. "
for file in *
do
if [ -f $file ]
then
if [ -r $file -a -w $file -a -x $file ]
then
ls -l $file
fi
fi
done
```

OUTPUT:

```
user1@user1:~$ sh week9a.sh
The list of File Names in the curent directory.
Which have Read,Write and Execute permisions.

-rwxrwxr-x 1 chand chand 0 Apr  2 10:06 XDG_VTNR=7
-rwxrwxr-x 1 chand chand 8728 Apr  2 10:30 a.out
-rwxrwxrwx 1 chand chand 497 Mar 30 12:05 armarg.sh
-rwxrwxrwx 1 chand chand 193 Apr  2 13:43 bc.sh
-rwxrwxrwx 1 chand chand 1460 Apr  2 10:50 binary.sh
-rwxrwxrwx 1 chand chand 661 Apr  1 11:14 binay.sh
-rwxrwxrwx 1 chand chand 727 Apr  2 10:41 bubble.sh
-rwxrwxrwx 1 chand chand 246 Apr  2 11:11 count.awk
-rwxrwxrwx 1 chand chand 581 Apr  1 11:57 demo.sh
-rwxrwxrwx 1 chand chand 66 Apr  2 11:02 ex
-rwxrwxrwx 1 chand chand 143 Apr  1 11:29 floop.sh
-rwxrwxrwx 1 chand chand 183 Apr  2 10:10 large.sh
-rwxrwxr-x 1 chand chand 220 Apr  2 13:01 maxofn.sh
-rwxrwxrwx 1 chand chand 438 Apr  2 21:40 range.sh
-rwxrwxrwx 1 chand chand 1237 Apr  2 10:43 select.sh
-rwxrwxrwx 1 chand chand 375 Apr  1 12:05 sink.sh
-rwxrwxrwx 1 chand chand 1346 Apr  2 10:35 week10.c
-rwxrwxr-x 1 chand chand 199 Apr  2 10:01 week11bn.c
-rwxrwxrwx 1 chand chand 236 Apr  2 10:27 week12n.c
-rwxrwxrwx 1 chand chand 669 Apr  2 21:37 week132.sh
```

9.

b) Develop an interactive script that ask for a word and a file name and then tells how many times that word occurred in the file.

```
echo  " Enter the word to be searched"
   read  word
   echo  "Enter the filename to be used"
   read  flname
   echo  "the no. of times the word occured in the file."
   grep  -c  $word  $flname
```

OUTPUT:

```
user1@user1:~$ sh week9b.sh
 Enter the word to be searched
echo
Enter the filename to be used
week8a.sh
the no. of times the word occured in the file.
24
```

9.

c) Write a shell script to perform the following string operations:

i)To extract a sub-string from a given string.

```
echo "Enter a string : "
read s
echo "Enter start pos : "
read m
echo "Enter end pos : "
read n
echo ${s:$m:$n}
```

OUTPUT:

```
Enter a string :
dadhirao chandrika
Enter start pos :
1
Enter end pos :
4
adhi
```

ii)To find the length of a given string.

```
echo "To find the length of the given string."
 echo "Enter the string."
 read string
 strlen=${#string}
 echo "The string length is : $strlen"
```

OUTPUT:

```
user1@user1:~$ sh week9c2.sh
To find the length of the given string.
Enter the string.
Chandrika Dadhirao
The string length is : 18
```

Program 10:

Write a C program that takes one or more file or directory names as command line input and reports the following information on the file:

i) File Type
ii) Number of links
iii) Read, write and execute permissions
iv) Time of last access

(Note: Use stat/fstat system calls)

```
OUT#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<time.h>
int main(int argc,char *argv[])
{
char *at,*mt,*ct;
struct stat buf;
stat(argv[1],&buf);
printf("the inode no is : %d\n",buf.st_ino);
if(S_ISDIR(buf.st_mode))
printf("it is a directory");
if(S_ISREG(buf.st_mode))
printf("it is regurlar file\n");
printf("the no of links is %d\n",buf.st_nlink);
at=ctime(&buf.st_atime);
printf("the time of last access is %s\n",at);
mt=ctime(&buf.st_mtime);
printf("the modification time is :%s\n",mt);
ct=ctime(&buf.st_ctime);
printf("time of last change is : %s\n",ct);
if((buf.st_mode&S_IRUSR)==S_IRUSR)
printf("user has read permission\n");
if((buf.st_mode &S_IWUSR)==S_IWUSR)
printf("user has write permisssion\n");
if((buf.st_mode &S_IXUSR)==S_IXUSR)
printf("user has execute permission\n");
if((buf.st_mode & S_IRGRP)==S_IRGRP)
printf("group has read permission\n");
if((buf.st_mode &S_IWGRP)==S_IWGRP)
printf("group has write permission\n");
if((buf.st_mode & S_IXGRP)==S_IXGRP)
printf("group has execute permission\n");
```

```
if((buf.st_mode &S_IROTH)==S_IROTH)
printf("others has read permission\n");
if((buf.st_mode &S_IWOTH)==S_IWOTH)
printf("others has read permission\n");
if((buf.st_mode &S_IWOTH)==S_IWOTH)
printf("others has write permission\n");
if((buf.st_mode &S_IXOTH)==S_IXOTH)
printf("others has execute permisssion\n");
}
```

OUTPUT:

```
user1@user1:~$ chmod 777 week10.c
user1@user1:~$ ./a.out week10.c week11an.c

the inode no is : 1966904

it is regurlar file

the no of links is 1
the time of last access is Thu Apr  2 10:30:23 2015
the modification time is :Thu Apr  2 10:30:17 2015
time of last change is : Thu Apr  2 10:30:35 2015

user has read permission
user has write permisssion
user has execute permission

group has read permission
group has write permission
group has execute permission

others have read permission
others have read permission
others have write permission
others have execute permisssion
```

Program 11:

Write C program that simulate the following unix commands:

a)mv

```
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
main( int argc,char *argv[] )
{
int i,fd1,fd2;
char *file1,*file2,buf[2];
file1=argv[1];
file2=argv[2];
printf("file1=%s file2=%s \n",file1,file2);
fd1=open(file1,O_RDONLY,0777);
fd2=creat(file2,0777);
while(i=read(fd1,buf,1)>0)
write(fd2,buf,1);
remove(file1);
close(fd1);
close(fd2);
}
```

OUTPUT:

```
user1@user1:~$ cc week11a.c
user1@user1:~$ ./a.out week11a.c week11an.c
file1=week11a.c file2=week11an.c
user1@user1:~$ cat week11a.c
cat: week11a.c: No such file or directory
```

b) cp (use system calls)

```
#include<stdio.h>
int main(int argc,char *argv[])
{
FILE *fp1,*fp2;
  int ch;
  fp1=fopen(argv[1],”r”);
fp2=fopen(argv[2],"w");
while((ch=fgetc(fp1))!=-1)
fputc(ch,fp2);
}
```

OUTPUT:

```
user1@user1:~$ chmod 777  week11b.c
user1@user1:~$ ./a.out week11b.c week11bn.c
file1=week11b.c file2=week11bn.c
user1@user1:~$ cat week11bn.c
#include<stdio.h>
int main(int argc,char *argv[])
{
        FILE *fp1,*fp2;
          int ch;
          fp1=fopen(argv[1],”r”);
fp2=fopen(argv[2],"w");
while((ch=fgetc(fp1))!=-1)
fputc(ch,fp2);
}
```

Program 12:

Write a C program that simulates ls command (Use system calls/directory API)

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<dirent.h>
int main(int argc,char *argv[])
{
DIR *dp;
struct dirent *sd; dp=opendir(argv[1]); while((sd=readdir(dp))!=NULL)
{
printf("%s\t",sd->d_name);
}
closedir(dp);
}
```

OUTPUT:

```
binay.sh week12n.c tst.sh .bash_logout lab16.sh 1tonarm.sh
.bash_history a.out .dbus Templates .local .dmrc
zerro.sh.ICEauthority .Xauthority .gstreamer-0.10 .xsession-
errors XDG_VTNR=7 Music arm1.sh sink.sh floop.sh
menu.sh lab14.awk Desktoplab sh.awk week11bn.c fz.sh .gconf
Public week11a.c tst1.sh .profileperfect.sh large.sh .xsession-
errors.old .compiz . demo..cache .bashrc week133.sh
Downloads Videos .lesshst imp week12.c armarg.sh
lab15.awk arm.sh week132.sh .. studentmarks student.dat
Pictures .mozilla Documents studentmarks.dat
```

Program 13:

Do the following Shell Programs also

1) Write a shell script to check whether a particular user has logged in or not. If he has logged in, also check whether he has eligibility to receive a message or not.

```
#get username
echo "enter username"
read username
who | grep $username
```

OUTPUT:

```
user1@user1:~$ sh week131.sh
enter username
chand
chand    :0          2015-04-02 09:42 (:0)
chand    pts/13      2015-04-02 09:43 (:0)
chand    pts/7       2015-04-02 12:12 (:0)
```

2) Write a shell script to accept the name of the file from standard input and perform the following tests on it.
a)File executable
b)File readable
c)File writable
d)Both readable & writable

```
while true
do
echo "
1. Executable file
2. Readable file
3. Writeable file
4. Both Readable and Writeable "
read choice
case $choice in
1) if [ -x $1 ]
then
echo "$1 is Executable file"
else
echo "$1 is not Executable file"
fi
```

```
;;
2) if [ -r $1 ]
then
echo "$1 is Readable file"
else
echo "$1 is not Readable file"
fi
;;
3) if [ -w $1 ]
then
echo "$1 is Writeable file"
else
echo "$1 is not Writeable file"
fi
;;
4) if [ -r $1 -a -w $1 ]
then
echo "$1 is Both readable and writable file"
else
echo "$1 is not readable and writable file"
fi
;;
*) Echo " Not a valid file"
;;
esac
echo " Do you want to continue (y) :"
read x
if [ $x = "n" ]
then
exit
fi
done
```

OUTPUT:

```
user1@user1:~$ ./week132.sh week133.sh

1. Executable file
2. Readable file
3. Writeable file
4. Both Readable and Writeable
1
week133.sh is not Executable file
 Do you want to continue (y) :
```

```
y

1. Executable file
2. Readable file
3. Writeable file
4. Both Readable and Writeable
2
week133.sh is Readable file
 Do you want to continue (y) :
y

1. Executable file
2. Readable file
3. Writeable file
4. Both Readable and Writeable
3
week133.sh is Writeable file
 Do you want to continue (y) :
y

1. Executable file
2. Readable file
3. Writeable file
4. Both Readable and Writeable
4
week133.sh is Both readable and writable file
 Do you want to continue (y) :
n
```

3) Write a shell script which all display the username and terminal name who login recently into the unix system.

```
who -s -H
NAME    LINE        TIME            COMMENT
chand   :0          2015-04-02 09:42 (:0)
chand   pts/13      2015-04-02 09:43 (:0)
chand   pts/7       2015-04-02 12:12 (:0)
```

4) Write a shell script to find no. of files in a directory

```
f=0
d=0
for fn in `ls *`
do
```

```
 if [ -d $fn ]
   then
     d=`expr $d + 1`
  else
     f=`expr $f + 1`
  fi
done
echo "No. of files : $f"
```

OUTPUT:

```
user1@user1:~$ sh week133.sh
No. of files : 22
```

5) Write a shell script to check whether a given number is perfect or not

```
echo Enter a number
read no
i=1
ans=0
while [ $i -le `expr $no / 2` ]
  do
     if [ `expr $no % $i` -eq 0 ]
     then
       ans=`expr $ans + $i`
     fi
     i=`expr $i + 1`
done
if [ $no -eq $ans ]
then
echo $no is perfect
else
echo $no is NOT perfect
fi
```

OUTPUT:

```
user1@user1:~$ sh perfect.sh
Enter a number
6
6 is perfect
user1@user1:~$ sh perfect.sh
Enter a number
7
7 is NOT perfect
```

6) Write a menu driven shell script to copy , edit, rename and delete a file

```
echo "Menu "
echo "1. Copy a File "
echo "2. Remove a file "
echo "3. Move a file"
echo "4. Quit"
echo "Enter ur Choice \c"
read Choice
case "$Choice" in
  1) echo "Enter File name to copy \c"
     read f1
     echo "Enter FIle name \c "
  read f2
  if [ -f $f1 ]
  then  cp $f1 $f2
     else
          echo "$f1 does not exist"
     fi
     ;;
  2) echo "Enter the File to be removed "
     read r1
     if [ -f $r1 ]
     then
          rm -i $r1
     else
         echo "$r1 file does not exist "
     fi
     ;;
  3)
     echo "Enter File name to move \c"
     read f1
     echo "Enter destination \c "
     read f2
     if [ -f $f1 ]
      then
         if [ -d $f2 ]
         then
             mv $f1 $f2
         fi
      else
echo "$f1 does not exist"
      fi
```

```
        ;;
    4)
        echo "Exit......."
        exit;;
esac
```

OUTPUT

```
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter ur Choice 4
Exit.....

user1@user1:~$ sh week136.sh
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter ur Choice 3

Enter File name to move: week133.sh
Enter destination: direct.sh
user1@user1:~$ cat direct.sh
cat: direct.sh: No such file or directory
user1@user1:~$ cat week133.sh
if [ -d "$@" ]
then
find "$@" -type f | ls -l "$@" | wc -l | echo "Number of files is $@":$@
find "$@" -type d | ls -l "$@" | wc -l | echo "Number of directories is $@":$@
fi

user1@user1:~$ sh week136.sh
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter ur Choice 2
```

```
Enter the File to be removed : week133.sh
rm: remove regular file ‘week133.sh’? y
user1@user1:~$ cat week133.sh
cat: week133.sh: No such file or directory
user1@user1:~$ sh week136.sh
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter ur Choice 1

Enter File name to copy week132.sh
Enter FIle name wee.sh
user1@user1:~$ cmp week132.sh wee.sh
```

7) Write a shell script for concatenation of two strings

```
echo  “Enter first string:”
read s1
echo “Enter second string:”
read s2
st=`echo $s1$s2`
#st=$s1$s2
echo "Without assignment : $s1$s2"
echo "Concatenated string is: $st"
```

OUTPUT:

```
user1@user1:~$ sh week137.sh
“Enter first string:”
chandrika
“Enter second string:”
dadhiao
Without assignment : chandrikadadhirao
Concatenated string is: chandrikadadhirao
```

8) Write a shell script which will display Fibonacci series up to a given number of arguments.

```
echo enter the number
read n
i=1
```

```
f=0
s=1
next=`expr $f + $s`
echo $f
echo $s
echo $next
while [ $i -le `expr $n - 3` ]
do
f=$s
s=$next
next=`expr $f + $s`
echo $next
i=`expr $i + 1`
done
```

OUTPUT:

```
user1@user1:~$ sh fibbo.sh
enter the number
20

0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
```

9) Write a shell script to accept student number, name, marks in 5 subjects. Find total, average and grade. Display the result of student and store in a file called stu.dat

Rules: avg>=80 then grade A
Avg<80&&Avg>=70 then grade B
Avg<70&&Avg>=60 then grade C
Avg<60&&Avg>=50 then grade D
Avg<50&&Avg>=40 then grade E
Else grade F

```
echo -n Enter Student Number :
read no
echo -n Enter Student Name :
read name
echo Enter Student Marks
echo -n Sub1 :
read s1
echo -n Sub2 :
read s2
echo -n Sub3 :
read s3
echo -n Sub4 :
read s4
echo -n Sub5 :
read s5
tot=`expr $s1 + $s2 + $s3 + $s4 + $s5`
echo Total Marks $tot
avg=`expr $tot / 5`
echo Average Marks $avg
if [ $s1 -ge 35 -a $s2 -ge 35 -a $s3 -ge 35 -a $s4 -ge 35 -a $s5 -ge 35 ]
then
echo "Pass"
if [ $avg -ge 81 ]
then
Grade="A"
elif [ $avg -ge 71 ]
then
Grade="B"
elif [ $avg -ge 61 ]
then
Grade="C"
elif [ $avg -ge 51 ]
```

```
then
Grade="D"
elif [ $avg -ge 41 ]
then
Grade="E"
else
Grade="F"
fi
echo Grade $Grade
else
echo "fail"
fi
echo $no $name $s1 $s2 $s3 $s4 $s5 $tot $avg $Grade >> stu.dat
```

OUTPUT:

```
user1@user1:~$ sh week139.sh
Enter Student Number :1203
Enter Student Name :anisha
Enter Student Marks
Sub1 :90
Sub2 :90
Sub3 :80
Sub4 :80
Sub5 :70
Total Marks 410
Average Marks 82
Pass
Grade A
user1@user1:~$ cat stu.dat
1206 chandrika 80 80 80 80 80 400 80 B
1206 chandrika 80 80 80 80 80 400 80 B
1203 anisha 90 90 80 80 70 410 82 A
```

10) Write a shell script to accept empno, emp name, basic. Find DA, HRA, TA, PF using following rules. Display empno, empname, basic, DA, HRA,PF,TA,GROSS SAL and NETSAL. Also store all details in a file calledemp.dat

Rules: HRA is 18% of basic if basic > 5000 otherwise 550

DA is 35% of basic

PF is 13% of basic

IT is 14% of basic

TA is 10% of basic

```
echo -n "Enter Employee Number : "
```

```
read empno
echo -n "Enter Employee Name : "
read empname
echo -n "Enter Basic Salary : "
read basic
if [ $basic -gt 5000 ]
then
hra=$((18*$basic/100))
else
hra=550
fi
da=$((35*$basic/100))
pf=$((12*$basic/100))
ta=$((10*$basic/100))
gross=$((da+hra+ta-pf))
netsal=$((basic+gross))
echo "----------------------"
echo " Employee Details "
echo "----------------------"
echo
echo "Employee No : $empno"
echo "employee name: $empname"
echo "Basic Salary : $basic"
echo "DA : $da"
echo "HRA : $hra"
echo "PF : $pf"
echo "TA : $ta"
echo "Gross Salary : $gross"
echo
echo "----------------------"
echo "Net Salary : $netsal"
echo "----------------------"
echo $empno $empname $basic $da $hra $pf $ta $gross $netsal >>emp.dat
```

OUTPUT:

```
user1@user1:~$ sh week1310.sh
Enter Employee Number : 1203
Enter Employee Name : rajiv
Enter Basic Salary : 4000
----------------------
 Employee Details
----------------------

Employee No : 1203
```

```
employee name: rajiv
Basic Salary : 4000
DA : 1400
HRA : 550
PF : 480
TA : 400
Gross Salary : 1870
---------------------
Net Salary : 5870
---------------------
user1@user1:~$ cat emp.dat
1206 chandika 8000 2800 1440 960 800 4080 12080
12 kuma 2347 821 550 281 234 1324 3671
1203 rajiv 4000 1400 550 480 400 1870 5870
```

11) Write a shell script to demonstrate break and continue statements

Break

```
a=0

while [ $a -lt 10 ]
do
  echo $a
  if [ $a -eq 5 ]
  then
    break
  fi
  a=`expr $a + 1`
done
```

OUTPUT:

```
vignan@vignan-desktop:~$ sh break.sh
0
1
2
3
4
5
```

Continue

```
NUMS="1 2 3 4 5 6 7"
```

```
for NUM in $NUMS
do
  Q=`expr $NUM % 2`
  if [ $Q -eq 0 ]
  then
    echo "Number is an even number!!"
    continue
  fi
  echo "Found odd number"
done
```

OUTPUT:

```
vignan@vignan-desktop:~$ sh continue.sh
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
```

12) Write a shell script to satisfy the following menu options
a. Display current directory path
b. Display todays date
c. Display users who are connected to the unix system.
d. Quit

```
while  true
do
 echo "
       1. current working directoy
       2. Today's date
       3. List of Users
       4. Quit"
 echo "Enter Your choice [1-4] :"
 read ch
 case $ch in
  1) pwd ;;
  2) date "+%d-%m-%Y" ;;
  3) who ;;
  4) exit ;;
  *) echo "Invalid  choice"
 esac
```

```
done
```

OUTPUT:

```
user1@user1:~$ sh menu.sh
        1. current working directoy
        2. Today's date
        3. List of Users
        4. Quit
Enter Your choice [1-4] :1
/home/chand

        1. current working directoy
        2. Today's date
        3. List of Users
        4. Quit
Enter Your choice [1-4] : 2
31-03-2015

        1. current working directoy
        2. Today's date
        3. List of Users
        4. Quit
Enter Your choice [1-4] : 3
chand    :0          2015-03-31 19:09 (:0)
chand    pts/0       2015-03-31 19:12 (:0)
```

13) Write a shell script to delete all files whose size is zero bytes from current directory

```
for fn in `ls *`
do
  if [ ! -s $fn -a ! -d $fn ] ; then
   echo "$fn - zero bytes "
   rm -i $fn
  fi
done
```

OUTPUT:

```
user1@user1:~$ sh fz.sh
ab - zero bytes
rm: remove regular empty file ‘ab’? y
bv - zero bytes
rm: remove regular empty file ‘bv’? y
cd - zero bytes
```

Program 14:

Write a shell script to display string palindrome from given arguments.

```
s=`echo $1 | wc -c`
while [ $s -gt 0 ]
do
temp=`echo $1 | cut -c $s`
s=`expr $s - 1`
temp1="$temp1$temp"
done
echo "Reverse string is $temp1"
if [ $1 = $temp1 ]
then
echo “The given string is palindrome”
else
echo “ Not palindrome”
fi
```

OUTPUT:

```
user1@user1:~$ sh palindrome.sh madam
Reverse string is madam
“The given string is palindrome”
user1@user1:~$ sh palindrome.sh mada
Reverse string is adam
“ Not palindrome”
```

Program 15:

Write a shell script which will display Armstrong numbers from given arguments.

```
echo "Enter a value : "
read n
t=$n
s=0
while [ $n -ne 0 ]
do
 d=`expr $n % 10`
 x=`expr $d \* $d \* $d`
 s=`expr $s + $x`
 n=`expr $n / 10`
done
if [ $s -eq $t ] ; then
 echo "$t is a armstong number "
else
 echo "$t is not a armstrong number "
fi
```

OUTPUT:

```
user1@user1:~$ sh arm.sh
Enter a value :
153
153 is a armstong number
user1@user1:~$ sh arm.sh
Enter a value :
35
35 is not a armstrong number
```

Program 16:

Write a shell script to display reverse numbers from given arguments list.

```
while [ $# -ne 0 ]
do
n=$1
while [ $n -ne 0 ]
do
r=$((n%10))
s=$((s*10+r))
n=$((n/10))
done
shift
echo "$s"
s=" "
done
```

OUTPUT:

```
user1@user1:~$ sh week1316.sh 123584379574
475973485321
```

Program 17:

Write a shell script to display factorial value from given argument list.

```
i=1
f=1
echo "Enter the number"
read n
while [ $i -le $n ]
do
f=`expr $f \* $i`
i=`expr $i + 1`
done
echo FACTORIAL = $f
```

OUTPUT:

```
user1@user1:~$ sh facto.sh
Enter the number
20
FACTORIAL = 2432902008176640000
```

Program 18:

Write a shell script which will find maximum file size in the given argument list.

```
vignan@vignan-desktop:~$ find . -type f -print0 | xargs -0 du | sort -n | tail -1 | cut -f2 |
xargs -I{} du -sh {}
317M
./.PlayOnLinux/ressources/WindowsXP-KB936929-SP3-x86-ENU.exe
```

Program 19:

Write a shell script which will greet you " Good Morning", "Good Afternoon", "Good Evening" and "Good Night" acording to current time.

```
x=`who am i | tr -s " " | cut -d " " -f4 | cut -c 1-2 `
if [ $x -ge 5 -a $x -lt 12 ]
then
echo "good morning"
elif [ $x -ge 12 -a $x -lt 16 ]
then
echo "good after"
elif [ $x -ge 16 -a $x -le 21 ]
then
echo "good evening"
fi
```

OUTPUT:

```
user1@user1:~$ sh greet.sh
good morning
user1@user1:~$ sh greet.sh
good evening
```

Program 20:

Write a shell script to sort the elements in a array using bubble sort technique.

```
printnumbers()
{
echo ${ARRAY[*]}
#You can also use bellow code
#for ((i=0;i<count;i++))
#do
#echo -n " ${ARRAY[i]} "
#done
}
exchange()
{
temp=${ARRAY[$1]}
ARRAY[$1]=${ARRAY[$2]}
ARRAY[$2]=$temp
}
sortnumbers()
{
for((last=count-1;last>0;last--))
do
   for((i=0;i<last;i++))
   do
     j=$((i+1))
     if [ ${ARRAY[i]} -gt ${ARRAY[j]} ]
     then
      exchange $i $j
     fi
   done
done
}
echo "Enter Numbers to be Sorted"
read -a ARRAY
count=${#ARRAY[@]}
echo "Numbers Before Sort:"
printnumbers
echo
sortnumbers
echo "Numbers After Sort: "
printnumbers
```

OUTPUT:

```
user1@user1:~$ vi +24 bubble.sh
user1@user1:~$ ./bubble.sh
Enter Numbers to be Sorted
5 6 1 3 7 8
Numbers Before Sort:
5 6 1 3 7 8
Numbers After Sort:
1 3 5 6 7 8
```

Program 21:

Write a shell script to find largest element in array.

```
echo "Enter Numbers "
read -a ARRAY
count=${#ARRAY[@]}
max=${ARRAY[0]}
for((i=1;i<count;i++))
do
   if [  $max -lt ${ARRAY[i]} ]
   then
     max=${ARRAY[i]}
   fi
done
echo "The maximum number : $max"
```

OUTPUT:

```
user1@user1:~$ ./maxofn.sh
Enter Numbers to be Sorted
5 7 -3 2 7 12
5 7 -3 2 7 12
The maximum number : 12
```

Program 22:

Write awk program to print sum, avg of students marks list.

```
BEGIN{total=0;avg=0;
print"Name Grp S1 S2 S3 S4 S5\tTotal AVerage Result"
print"------- --- -- -- -- -- --\t ----- ---------- ------"}
{
total=$3+$4+$5+$6+$7;
avg=total/5;
if($3>35 && $4>35 && $5>35 && $6>35 && $7>35)
{
{print $0 "\t" total "\t" avg "\t" "PASS" }
}
else
{
{print $0 "\t" total "\t" avg "\t" "Fail" }
}
}
END{}
```

OUTPUT:

```
$ cat student.data
        Pavan MSC  55   66    77    88    99
        kumar      MSC  65   48    75    95    75
        Raju  MSC  78   95    78    35    78
        Bala  MSC  85   48    55    95    75
        chakri MSC  51   25    48    35    39

$ awk -f lab14.awk student.data

Name        Grp  S1   S2   S3   S4   S5   Total  Average  Result
-------------------------------------------------------------------------------------------

Pavan       MSC  55   66   77   88   99   385    77       PASS
kumar       MSC  65   48   75   95   75   358    71.6
        PASS
Raju        MSC  78   95   78   35   78   364    72.8     Fail
Bala        MSC  85   48   55   95   75   358    71.6     PASS
Chakri      MSC  51   25   48   35   39   198    39.6
        Fail
```

Program 23:

Write awk program to display students pass/fail report.

```
vi lab.awk
{
if ($3 >=35 && $4 >= 35 && $5 >= 35)
        print $0,"=>","Pass";
else
        print $0,"=>","Fail";

}
cat studentmarks.dat
Jones 2143 78 84 77 => Pass
Gondrol 2321 56 58 45 => Pass
RinRao 2122 38 37 => Fail
Edwin 2537 87 97 95 => Pass
Dayan 2415 30 47 => Fail
```

OUTPUT:

```
user1@user1:~$ awk -f lab.awk studentmarks.dat
Jones 2143 78 84 77 => Pass => Pass
Gondrol 2321 56 58 45 => Pass => Pass
RinRao 2122 38 37 => Fail => Pass
Edwin 2537 87 97 95 => Pass => Pass
Dayan 2415 30 47 => Fail => Fail
```

Program 24:

Write a awk program to count number of vowels in a given file.

```
BEGIN{print "record.\t characters \t words"}
#BODY section
{
len=length($0)
total_len+=len
print(NR,":\t",len,":\t",NF,$0)
words+=NF
}
END{
print("\n total")
print("characters :\t" total_len)
print("words :\t"words)
print("lines :\t" NR)
}
```

OUTPUT:

```
-bash-3.2$ cat ex
this is an example of awk
counting vowels
not containing in lines
```

awk –f count.awk ex

record. characters words

1 : 25 : 6 this is an example of awk

2 : 15 : 2 counting vowels

3 : 23 : 4 not containing in lines

total

characters : 63

words: 12

lines : 3

Program 25:

Write a awk program which will find maximum word and its length in the given input file.

```
 maxlength=0;
  maxword=0
}
{
  for(i=1;i<=NF;i++)
   if (length($i)>maxlength)
   {
    maxlength=length($i);
    maxword=$i;
   }
}
END {
  print maxword,maxlength;
}
```

OUTPUT:

```
vignan@vignan-desktop:~$ awk -f maxword.awk hi
chandrikadadhirao 17
```

Program 26:

Write a shell script to generate the mathematical tables.

```
echo Multiplication Table
echo which table do u want
read num
echo enter range
read r
for((i=0;i<=$r;i++))
do
res=`expr $num \* $i`
  $num "*" $i "=" $res
done
```

OUTPUT:

```
Multiplication Table
which table do u want
3
enter range
5
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
```

Program 27:

Write a shell script to sort elements of given array by using selection sort.

```
printnumbers()
{
echo ${ARRAY[*]}
}

swap()
{
temp=${ARRAY[$1]}
ARRAY[$1]=${ARRAY[$2]}
ARRAY[$2]=$temp
}

sortnumbers()
{
for ((i=0;i<count;i++))
do
min=$i
  for ((j=i+1;j<count;j++))
  do
     if [ ${ARRAY[j]} -lt ${ARRAY[min]} ]
     then
     min=$j
     fi
  done
  swap $i $min
done
}
echo "Enter Numbers to be Sorted : "
read -a ARRAY
count=${#ARRAY[@]}
echo "Numbers Before Sort:"
printnumbers
sortnumbers
echo "Numbers After Sort: "
printnumbers
binary search
printnumbers()
{
echo ${ARRAY[*]}
```

```
}
swap()
{
temp=${ARRAY[$1]}
ARRAY[$1]=${ARRAY[$2]}
ARRAY[$2]=$temp
}

sortnumbers()
{
for ((i=0;i<count;i++))
do
min=$i
 for ((j=i+1;j<count;j++))
 do
   if [ ${ARRAY[j]} -lt ${ARRAY[min]} ]
   then
   min=$j
   fi
 done
 swap $i $min
done
}
echo "Enter Numbers to be Sorted : "
read -a ARRAY
count=${#ARRAY[@]}
echo "Numbers Before Sort:"
printnumbers
sortnumbers
echo "Numbers After Sort: "
printnumbers
```

OUTPUT:

```
user1@user1:~$ chmod 777 select.sh
user1@user1:~$ ./select.sh
Enter Numbers to be Sorted :
4 6 1 0 3 -5
Numbers Before Sort:
4 6 1 0 3 -5
Numbers After Sort:
-5 0 1 3 4 6
./select.sh: line 55: binary: command not found
```

Enter Numbers to be Sorted :
1 2 6 0 7 8
Numbers Before Sort:
1 2 6 0 7 8
Numbers After Sort:
0 1 2 6 7 8

Program 28:

Write a shell script to search given number using binary search.

```
Ddb hbprintnumbers()
{
 echo ${ARRAY[*]}
}

sortnumbers()                    # Using insertion sort
{
for((i=1;i<count;i++))
do
  Temp=${ARRAY[i]}
  j=$((i-1))
  while [ $Temp -lt ${ARRAY[j]} ]
  do
    ARRAY[j+1]=${ARRAY[j]}
    let j--
    if [ $j == -1 ]
    then
      break
    fi
  done
  ARRAY[j+1]=$Temp
done
}

binarysearch()
{
   status=-1
   i=1
   array=($(echo "$@"))
   LowIndex=0
   HeighIndex=$((${#array[@]}-1))
   while [ $LowIndex -le $HeighIndex ]
   do
      MidIndex=$(($LowIndex+($HeighIndex-$LowIndex)/2))
      MidElement=${array[$MidIndex]}
      if [ $MidElement -eq $SearchedItem ]
      then
         status=0
         searches=$i
```

```
            return
        elif [ $SearchedItem -lt $MidElement ]
        then
            HeighIndex=$(($MidIndex-1))
        else
            LowIndex=$(($MidIndex+1))
        fi
        let i++
    done
}
clear
echo "Enter Array Elements : "
read -a ARRAY
count=${#ARRAY[@]}
search=y
sortnumbers
echo "Array Elements After Sort: "
printnumbers
while [ "$search" == "y" -o "$search" == "Y" ]
do
    echo -n "Enter Element to be searched : "
    read SearchedItem
    binarysearch "${ARRAY[@]}"
    if [ $status -eq 0 ]
    then
      echo "$SearchedItem found after $searches searches"
    else
       echo "$SearchedItem not found in the list"
    fi
    echo -n "Do you want another search (y/n): "
   read search
done
```

OUTPUT:

```
Enter Array Elements :
5 3 2 30 45 69
Array Elements After Sort:
2 3 5 30 45 69
Enter Element to be searched : 30
30 found after 3 searches
Do you want another search (y/n): y
Enter Element to be searched : 25
25 not found in the list
```

Do you want another search (y/n): n

Program 29:

Write a shell script to find number of vowels, consonants, numbers, whitespaces and special characters in agiven string.

```
echo "enter word"
read word
vowels=$(echo $word | sed 's/[^aeiou]//g')
consonants=$(echo $word | sed 's/[^bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]//g')
numbers=$(echo $numbers | sed 's/[^0123456789]//g')
space=$(echo $space | sed 's/[^" "]//g')
special=$(echo $word | sed 's/[^*%$#@!~`+_-=,.<>?]//g')
echo "${#word} characters"
rd=$word
consonants=$(echo $word | sed 's/[^bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]//g')
numbers=$(echo $word | sed 's/[^0123456789]//g')
space=$(echo $word | sed 's/[^" "]//g')
special=$(echo $word | sed 's/[^*%$#@!~`+_-=,.<>?]//g')
echo "${#word} characters"
echo "${#vowels} vowels"
echo "${#consonants} consonants"
echo "${#numbers} numbers"
echo "${#space} space"
echo "${#special} special"
```

OUTPUT:

```
user1@user1:~$ sh vcd1.sh
enter word
chandrika Dadhirao @ 31
23 characters
23 characters
7 vowels
10 consonants
2 numbers
3 space
1 special
```

Program 30:

Write a shell script to lock the terminal.

```
clear
stty -echo
echo "enter password to lock the terminal"
read pass1
echo " Re-enter password"
read pass2
if [ "$pass1" = "$pass2" ]
then
echo "system is locked"
echo "enter password to unlock"
trap ``/1 2 3 9 15 18
while true
do
read pass3
if [ $pass1 = $pass3 ]
then echo "system unlocked"
stty echo
exit
else
echo "password mismatch"
fi
done
else
echo "password mismatch"
stty echo
fi
```

OUTPUT:

```
enter password to lock the terminal

 Re-enter password
system is locked

enter password to unlock
system unlocked
```