**1(a)Study of Unix/Linux general purpose utility command list: man, who, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, echo, more, date, time, kill, history, chmod, chown, finger, pwd, cal, logout, shutdown.**

**Aim**: to purpose utility commands list

**1.man command:*man*** command in Linux is used to display the user manual of any command that we can run on the terminal.

1.**Syntax:** man [option(s)] keyword(s)

For example,

-a, --all

same as -b -d --login -p -r -t -T -u

-b, --boot

time of last system boot

-d, --dead

print dead processes

-H, --heading

print line of column headings

-l, --login

print system login processes

--lookup

attempt to canonicalize hostnames via DNS

-m only hostname and user associated with stdin

-p, --process

print active processes spawned by init

-q, --count

all login names and number of users logged on

-r, -- runlevel

print current runlevel

-s, --short

print only name, line, and time (default)

-t, --time

print last system clock change

-T, -w, --mesg

add user's message status as +, - or ?

-u, --users

list users logged in

Manual page who(1) line 1

**2.who command:** it prints the information who currently logged on.

**Syntax:**who[option]….[file|Arg1 Arg2]

[20A91A05CSE@Linux ~]$ who

exam41 pts/3 2021-10-27 09:16 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0502 pts/4 2021-10-27 11:09 (172-7-139-

250.lightspeed.irvnca.sbcglobal.net)

20A91A0551 pts/5 2021-10-27 10:18 (172-7-139-

250.lightspeed.irvnca.sbcglobal.net)

20A91A0513 pts/6 2021-10-27 09:52 (172-7-139-

250.lightspeed.irvnca.sbcglobal.net)

20A91A0547 pts/7 2021-10-27 09:54 (172-7-139-

250lightspeed.irvnca.sbcglobal.net

**3.cat command:** the purpose of this command is

a.to create a file

b.to display the content of the file

c.to append a new content in the file existing content in a file

**a**. To create a file syntax:: cat><file name> example:

[20A91A05CSE@linux~]$ cat>file1.txt

Welcome

To unix and shell programming lab

**b**. To display the content of the file syntax:: cat <filename.extension> example:

[20A91A05CSE@linux~]$ cat file1.txt

Welcome

To unix and shell programming lab

**c.**to append a new content in the file existing content in a file

syntax: cat>><filename.extension>

example:

[20A91A05CSE@linux~]$ cat>FILE

Aditya engineering college

[ 20A91A05CSE@linux~]$ cat sushma

Welcome

To unix and shell programming lab

Aditya engineering college

**4.cd:** to change the directory

Syntax: cd<directoryname> Example:

[20A91A05CSE@linux~]$mkdir file

[20A91A05CSE@linux~]$ cd file

[20A91A05CSE@linux file]$

**5.cp:** it copies the content from source file to destination file.

Syntax: cp<sourcefilename><destination filename> Example:

[20A91A05CSE@linux~]$ cp file1 file2

[20A91A05CSE@linux~]$ cat file2

Welcome

Aditya engineering college

**6.ps:** short for process is a command line utility that is used to display

Or view information related to the process running in a liunx system.

Syntax::ps

[20A91A05CSE@linux~]$ps

Pio tty time cmd

25740 pts/7 0: 00:00 bash

29001 pts/57 0:00:00 ps

**7) ls :**

ls it displays the list directory contents

Syntax: ls

Example

[20A91A05CSE@linux~]$ ls

add.ccosx.cfile.cabc.c

**8)mv:**

it moves the content from the source file to distination file

Syntax: mv <sourcefile><destination filename>

Example

[20A91A05CSE@linux~]$ mv file1 file2

[20A91A05CSE@linux~]$ cat file2

Welcome

To unix and shell programming lab

Aditya engineering college

**9.rm**: it is used to remove the files

Syntax: rm<filename>

Example

[20A91A05CSE@linux~]$rm file2

**10.mkdir:** it is used to create a directory

Syntax: mkdir<directory name>

Example

[20A91A05CSE@linux~]$mkdirkmss

**11.rmdir:** to remove a directory

Syntax: rm<directory name>

[20A91A05CSE@linux~]$rmdirkmss

[20A91A05CSE@linux~]$

**12.echo:** it is used to display the line of text/string that are passed as an argumanet

Syntax: echo example

[20A91A05CSE@linux~]$ echo

Hello hello”echo”world

World

**13.more:** it is used to view the text files in command prompt displaying one screen at a time in case the file is large

Syntax:

More[-options] [-num][\*/pattern][\*line-num][filename] example

[20A91A05CSE@linux~]$ more

**14.date:** it display the system date and time

Syntax: date

Example: [20A91A05CSE@linux~]$ date

Wed oct 27 10:16:27 ist 2021

**15.kill:** it is used to terminate the process manually.

Syntax: kill -1

Example: [20A91A05CSE@linux~]$ kill -1

Sizhupsizo

Sizabrisizpwr

Siguitsizrtmay

**16.history:** it is used to view the previously executed command

Syntax: history Example:

[20A91A05CSE@linux~]$ history

Dear

Ln-s

Clear

Echo

**17.chmod**: it is used to the access mode of file

Syntax: chmod [refernce] [operator][mode] file…..

[20A91A05CSE@linux~]$chmodo+w list text

[20A91A05CSE@linux~]$ ls-1

Total 12 **:**

Dr xr-xt-xt-x-2 root root 4086 dec 31 07:16 text

Syntax: chown[option]…[owner][:[group]file

Example

[20A91A05CSE@linux~]$ ls-1 file.txt

-rw-r…r-1 root 66 feb 4 20:85 list text…

[20A91A05CSE@linux~]$chown master file.txt

Dr xr-xt-xt-x-2 root root 4086 feb 12 07:16 file. text

[20A91A05CSE@linux~]$

**19.finger:** it gives the details of all the users logged in

**Syntax :**finger<user name>

[20A91A05CSE@linux~]$ ls-1 finger bala

Example

Login balaname:bala

Directory: /home/sai/bala/ shell :/bin/bash

On since sun may 3 20:32 on: 0 from 0

No mail

No plan

[20A91A05CSE@linux~]$

**20.pwd:** it shows the present working directory

Syntax: pwd Example:

[20A91A05CSE@linux~]$cd cse

[20A91A05CSE@linux~]$pwd /home/20A91A05CSE/sai/cse

**21.cal:** it displays the calender

Syntax: cal Example:

[20A91A05CSE@linux~]$cal

October 2021

Su Mo Tue We Thu Fr Sa

1 2

3 4 5 6 7 8 9

10 11 12 13 14 15 16

17 18 19 20 21 22 23

16 25 26 27 28 29 30

31

**22.logout:** it allows you to programmatically logout from your session.

Syntax: session-logout[options]

Example:

To logout from the current session

[20A91A05CSE@linux~]$ logout

No output on the screen

**23.shutdown:** it is used to shutdown the system in a safe way only root

user can execute shutdown command.

Syntax: shutdown[optons][time][message]

[20A91A05CSE@linux~]$sudo +10”system”

Shutdown scheduled for sat 2021 04:20

15:43:06 iDT use,’ shutdown-c’ to cancel

[20A91A05CSE@linux~]$ shutdown-c

**16.time:**it is used to execute a command and prints a summary of real time,usercpu time and system cpu time spent by executing a command when it terminates

**Syntax:**time[option]

**Example:**[20A91A05CSE@linux~]$time

Real om3.003s

**1 b) Study of Bash shell, Bourne shell and C shell in Unix/Linux operating**

system.

**Introduction to Shells:**

➢ Shell is an environment in which we can run our commands, programs, and shell scripts. There are

different types of a shell, like different types of operating systems. Each shell has its own set of

recognized commands and functions.

➢ Every Unix system has at least one shell. A shell is a program that consists on the kernal and acts as

an agent or interface between the users and the kernal and hence the hardware. It is similar to the

command.com in the MS-DOS environment.

➢ A Shell provides user with an interface to the Unix system. It gathers input from user and executes

programs based on that input. When a program finishes executing, it displays that program&#39;s output.

➢ A Shell is a command interpreter or a processor .As the system is booted successfully , the shell

presents a command the prompt ($ or % symbols) at which the user can type in any unix command.

➢ After accepting the command , the shell generates a readily executable simple command line by

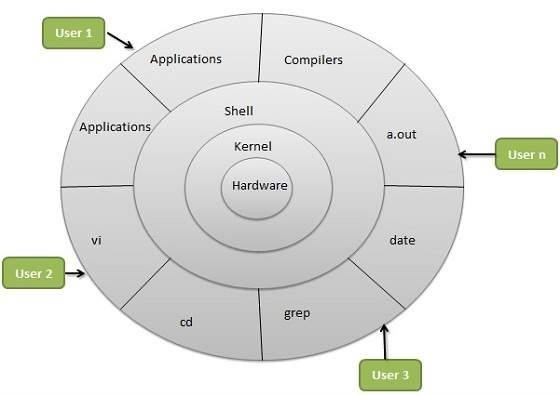
parsing it, evaluating variables, performs command substitution, interprets meta characters like \*

and ? and identifies the PATH.

➢ The prompt, $, which is called the command prompt, is issued by the shell. While the prompt is

displayed, you can type a command.

➢ Example :$date



**Types of Shells**

• Bourne shell − If you are using a Bourne-type shell, the $ character is the default prompt.

• The Bourne Shell has the following sub categories −

• Bourne shell (sh)

• Korn shell (ksh)

ROLL NO -20A91A05A4

• Bourne Again shell (bash)

• POSIX shell (sh)

• C shell − If you are using a C-type shell, the % character is the default prompt.

• The different C-type shells follow −

• C shell (csh)

TENEX/TOPS C shell (tcsh)

• Bourne shell (sh) :This is the most common shell available on Unix systems and the first major

shell to be developed . The shell is widely used. It has been named its author Stephen Bourne at

AT&amp;T Bell labs. This shell is distributed as the standard shell on all Unix systems.

• Korn shell (ksh) :Itwas developed by David Korn at AT&amp;T Bell labs. Basically it is built on the

Bourne shell. It is one of the widely used shells. It is freely available.

• Bourne Again shell (bash) :It was developed by B Fox and C Ramey at free software foundations.

Certain Linux os variants come with this shell as its default .

C shell − Bill Joy created it at the University of California at Berkeley. It is called C shell because its

syntax and usage is very similar to the C Programming language. Unfortunately this shell is not available

UNIX contains a system variable, SHELL that identifies the path to your login shell. We can check it with

the command as follows: Finding shell

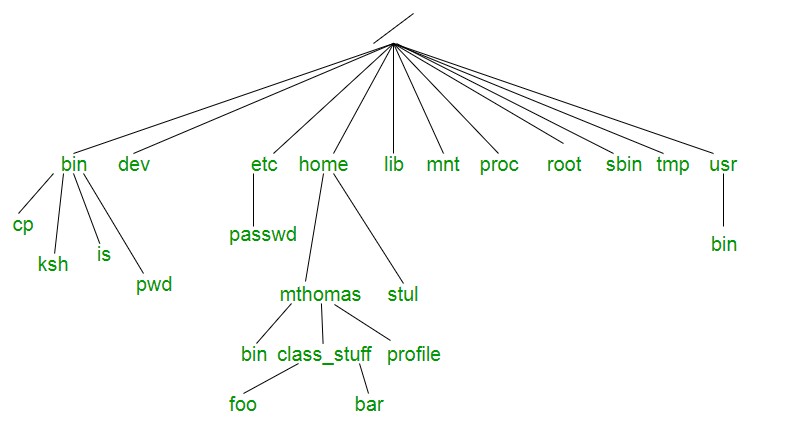
• $ echo $SHELL.

• $ echo $0

$ cat /etc/shells

ROLL NO -20A91A05A4

**1.c) Study of Unix/Linux file system (tree structure).**



A file system is a logical collection of files on a partition or disk UNIX uses a hierarchical file system

structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories

spreading from there.

A UNIX filesystem is a collection of files and directories that has the following properties :

1. It has a root directory (/) that contains other files and directories.

2. Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique

identifier, typically called an inode.

• By convention, the root directory has an inode number of 2 and the lost+found directory has an

inode number of 3. Inode numbers 0 and 1 are not used. File inode numbers can be seen by

specifying the -i option to ls command.

• It is self contained. There are no dependencies between one filesystem and any other. The

directories have specific purposes and generally hold the same types of information for easily

locating files. Following are the directories that exist on the major versions of Unix :

Directory Description

/ This is the root directory which should contain only the directories needed at

the top level of the file structure

/bin This is where the executable files are located. They are available to all user.

/dev These are device drivers.

/etc Supervisor directory commands, configuration files, disk configuration files,

valid user lists, groups, ethernet, hosts, where to send critical messages

**Directory Description**

/lib Contains shared library files and sometimes other kernel-related files

ROLL NO -20A91A05A4

/boot Contains files for booting the system.

/home Contains the home directory for users and other accounts

/mnt Used to mount other temporary file systems, such as cdrom and floppy for the

CDROM drive and floppy diskette drive, respectively

/proc Contains all processes marked as a file by process number or other

information that is dynamic to the system

/tmp Holds temporary files used between system boots

/user Used for miscellaneous purposes, or can be used by many users. Includes

administrative commands, shared files, library files, and others

/var Typically contains variable-length files such as log and print files and any

other type of file that may contain a variable amount of data

/sbin Contains binary (executable) files, usually for system administration. For

examplefdisk and ifconfig utlities.

/kernel Contains kernel files

**1.d) Study of .bashrc, /etc/bashrc and Environment variables.**

AIM: Study of .bashrc, /etc/bashrc and Environment variables

• The /etc/bashrc is executed for both interactive and non-interactive shells. /etc/bashrc or

/etc/bash.bashrc is the systemwide bash per-interactive-shell startup file. Is is used system wide

functions and aliases. However, environment stuff goes in /etc/profile file.the /etc/profile is

executed only for interactive shells

• .bashrc is a shell script that Bash runs whenever it is started interactively. It initializes an interactive

shell session.

• .bashrc runs on every interactive shell launch.

• UNIX contains a system variable, SHELL that identifies the path to your login shell. We can check

it with the command as follows:

Following is the partial list of important environment variables :-

• DISPLAY :Contains the identifier for the display that X11 programs should use by default.

• HOME :Indicates the home directory of the current user: the default argument for the cd built-in

command.

• IFS :Indicates the Internal Field Separator that is used by the parser for word splitting after

expansion.

ROLL NO -20A91A05A4

• LANG :LANG expands to the default system locale; LC\_ALL can be used to override this. For

example, if its value is pt\_BR, then the language is set to (Brazilian) Portuguese and the locale to

Brazil.

• LD\_LIBRARY\_PATH :On many Unix systems with a dynamic linker, contains a colonseparated

list of directories that the dynamic linker should search for shared objects when building a process

image after exec, before searching in any other directories.

• PATH :Indicates search path for commands. It is a colon-separated list of directories in which the

shell looks for commands.

• PWD :Indicates the current working directory as set by the cd command.

• RANDOM :Generates a random integer between 0 and 32,767 each time it is referenced.

• SHLVL :Increments by one each time an instance of bash is started. This variable is useful for

determining whether the built-in exit command ends the current session.

• TERM :Refers to the display type

• VZ :Refers to Time zone. It can take values like GMT, AST, etc.

• UID :Expands to the numeric user ID of the current user, initialized at shell startup.

**2.EXPERIMENT**

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

tabs to separatethe fields.

1425 Ravi 15.65

4160 Ramu 26.27

6830 Sita 36.15

1450 Raju 21.86

b) Study of vi editor

c) Use the cat command to display the file, my table.

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

e) Use the sort command to sort the file my table according to the first field. Call the

sorted file my table (same name).

f) Print the file my table.

**2a)create a table using cat command**

[20A91A05CSE@Linux ~]$ cat&gt;mytable

1425 Ravi 15.65

4160 Ramu 26.27

6830 Sita 36.15

1450 Raju 21.86

**2.b)Study of vi editor**

Aim: To Study of vi editor

vi is generally considered the de facto standard in Unix editors because −

• It&#39;s usually available on all the flavours of unix system.

• Its implementations are very similar across the board.

• It requires very few resources.

• It is more user-friendly than other editors such as the ed or the ex.

You can use the vi editor to edit an existing file or to create a new file from scratch. You

can also use this editor to just read a text file.

Syntax: vi filename

.vi editor has three modes

1)command mode

2)insert mode

3)exit mode

1)Command mode:

Once a file is open you are in the command mode .From command mode you can:

 Invoke insert mode

 Issue editing commands

ROLL NO -20A91A05A4

 Move cursor to a different position in the file

 Save and exit the current version of file

2)Insert mode:

In insert mode you can enter new text in the file press esc key to exit insert mode and

return to command mode.

The following commands invoke the insert mode:

 a Append after cursor

A Append at the end of line

 i Insert before cursor

 l Insert at beginning of line

 r Replace character under cursor

 Open a newline above current line

3)Lastline mode:

The last vi mode is known as vi last line mode. The following command invoke exit mode.

 :q to quit (short for quit)

 :q! to quit without saving

 :wq to write and quit

 :wq! To write and quit even if file has only read permission

 X to read and quit

 :qa to quit all (short for :quit all)

Example:[20A91A05532linux~]vi factorial.sh

**2c) display the table using cat command**

[20A91A05CSE@Linux ~] $ cat mytable

1425 Ravi 15.65

4160 Ramu 26.27

6830 Sita 36.15

1450 Raju 21.86

**2d)use vi command to edit**

[20A91A05CSE@Linux kmss]$ vi mytable

**2e)use sort command to sort**

[20A91A05CSE@Linux ~]$ sort -f +0 -1 mytable&gt;new.txt

[20A91A05CSE@Linux ~]$ cat new.txt

ROLL NO -20A91A05A4

1425 Ravi 15.65

1450 Raju 21.86

4160 Ramu 26.27

6830 Sita 36.15

**2f) print file my table**

[20A91A05CSE@Linux ~]$ cat mytable

1425 Ravi 15.65

4160 Ramu 26.27

6830 Sita 36.15

1450 Raju 21.86

**3.EXPERIMENT**

a) use the appropriate command to determine your login shell.

b) use the who command and redirect result to the file called myfile1,use the more

command to see the content of myfile1.

c) use the date and who command in sequence such that the output of date command

will display on the screen and the output of who command is redirected to a file called

myfile 2.

d)use the more command to check the content of myfile2.

ROLL NO -20A91A05A4

**a) to determine login shell**

[20A91A05CSE@Linux ~]$ echo $SHELL

/bin/bash

**b)who command and more command redirect to my file 1**

[20A91A05A4@Linux ~]$ who &gt;myfile1

[20A91A05A4@Linux ~]$ cat myfile1

20A91A0533 pts/0 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0534 pts/1 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0510 pts/2 2021-10-27 09:55 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

exam41 pts/3 2021-10-27 09:53 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

[20A91A05A4@Linux ~]$ more myfile1

20A91A0533 pts/0 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0534 pts/1 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0510 pts/2 2021-10-27 09:55 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

exam41 pts/3 2021-10-27 09:53 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

**c)date and who command on same file**

[20A91A05A4@Linux ~] $ date; who &gt;myfile2

Wed Oct 27 11:20:57 IST 2021

[20A91A05A4@Linux ~] $ cat myfile2

20A91A0533 pts/0 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0534 pts/1 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0510 pts/2 2021-10-27 09:55 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

ROLL NO -20A91A05A4

**d)more command to check content in myfile 2**

[20A91A05A4@Linux ~] $ more myfile2

20A91A0533 pts/0 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0534 pts/1 2021-10-27 09:42 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

20A91A0510 pts/2 2021-10-27 09:55 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

exam41 pts/3 2021-10-27 09:53 (172-7-139-250.lightspeed.irvnca.sbcglobal.net)

**4a**)Write a Shell Script that takes a Command-line argument and reports on whether it is directory,a file or something else

**AIM:** A Shell Script that takes a Command-line argument and reports on whether it is directory,a file or something else

**PROGRAM:**

if [ -f $1 ]

then

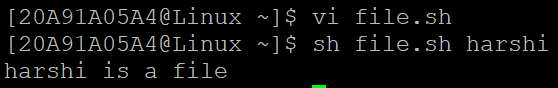
echo "$1 is a file"

else

echo "$1 is a Directory"

fi

**OUTPUT:**

****

**4b**) Write a Shell script that accepts one or more filename as arguments and converts all of them to uppercase provided they exist in the current directory

**AIM:** A Shell script that accepts one or more filename as arguments and converts all of them to uppercase provided they exist in the current directory

**PROGRAM:**

if [ -f $1 ]

then

echo "$1 is a file"

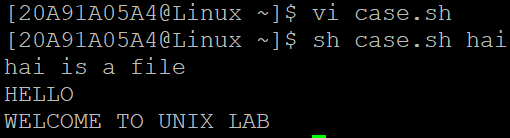
tr '[a-z]' '[A-Z]' <$1

else

echo "$1 is not a file"

fi

**OUTPUT:**



**5a**)Write a Shell script that determines the period for which a specified user is working on the system

**AIM:** A Shell script that determines the period for which a specified user is working on the system

**PROGRAM:**

echo "Enter the login of the User:"

read name

logindetails=`who|grep -w $name|grep 'pts'`

if [ $? -ne 0 ]

then

echo "$name has not logged in yet"

exit

fi

loginhours=`echo "$logindetails"|cut -c 26,27`

loginminutes=`echo "$logindetails"|cut -c 29-30`

hournow=`date|cut -c 12,13`

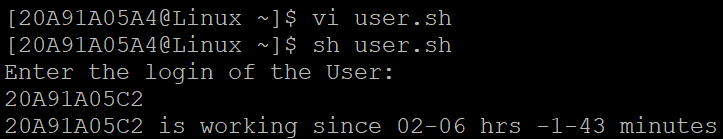
minnow=`date|cut -c 15,16`

hour=`expr $loginhours-$hournow`

min=`expr $loginminutes-$minnow`

echo "$name is working since $hour hrs $min minutes"

**OUTPUT:**



**5b**)Write a Shell script that accepts a Filename starting and ending line numbers as Arguments and displays all the lines between the given line numbers

**AIM:** A Shell script that accepts a Filename starting and ending line numbers as Arguments and displays all the lines between the given line numbers

**PROGRAM:**

echo "Enter the Filename"

read fname

echo "Enter starting line no"

read s

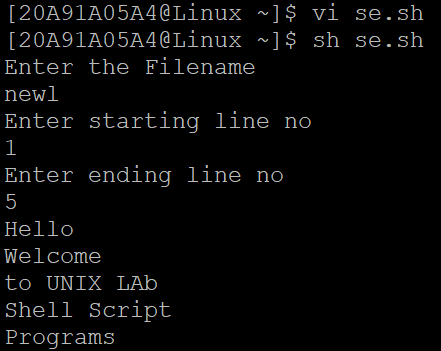
echo "Enter ending line no"

read n

sed -n $s,$n\p $fname | cat>newline

cat newline

**OUTPUT:**



**6)**Write a Shell Script that computes the Gross Salary of an Employee according to the following rules:

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

ii)If Basic Salary is >=1500 then HRA =Rs 500 and DA =98% of the Basic. The Basic Salary is entered actively through the Keyboard

**PROGRAM:**

echo "Enter the Basic Salary"

read bs

if [ $bs -lt 1500 ]

then

hra=`echo $bs\\*10/100|bc`

da=`echo $bs\\*90/100|bc`

else

hra=500

da=`echo $bs\\*98/100|bc`

fi

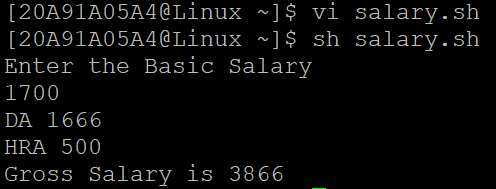
gs=`echo $bs+$hra+$da|bc`

echo "DA $da"

echo "HRA $hra"

echo "Gross Salary is $gs "

**OUTPUT:**



**7a**)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:** 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

**PROGRAM:**

if [ $# -ne 2 ]

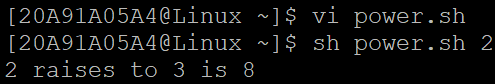
then

echo "Invalid no of Arguments"

fi

pwr=`echo $1^$2|bc`

echo "$1 raises to $2 is $pwr"

**OUTPUT:**

**7b**)Write a Shell Script which will display Armstrong Number from given Arguments

**AIM:** A Shell Script which will display Armstrong Number from given Arguments

**PROGRAM:**

for n in $\*

do

t=$n

sum=0

while [ $n -ne 0 ]

do

r=`expr $n % 10`

sum=`expr $sum + $r \\* $r \\* $r`

n=`expr $n / 10`

done

if [ $t -eq $sum ]

then

echo $t is an armstrong number

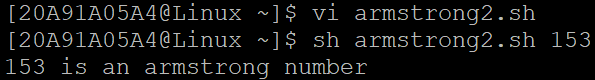
else

echo $t is not aarmstrong number

fi

done

**OUTPUT:**



**8**)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.

**PROGRAM:**

echo 1.copy

echo 2.rename

echo 3.remove

echo 4.link

echo 5.exit

echo "Enter Your Choice"

read ch

case $ch in

1)echo "Enter Source File"

read s

echo "Enter the Destination File"

read d

cp $s $d

;;

2)echo "Enter Old File Name"

read of

echo "Enter Neew File Name"

read nf

mv $of $nf

;;

3)echo "Enter the Filename to delete"

read df

rm $df

;;

4)echo "Enter File1"

read f1

echo "Enter File2"

read f2

ln $f1 $f2

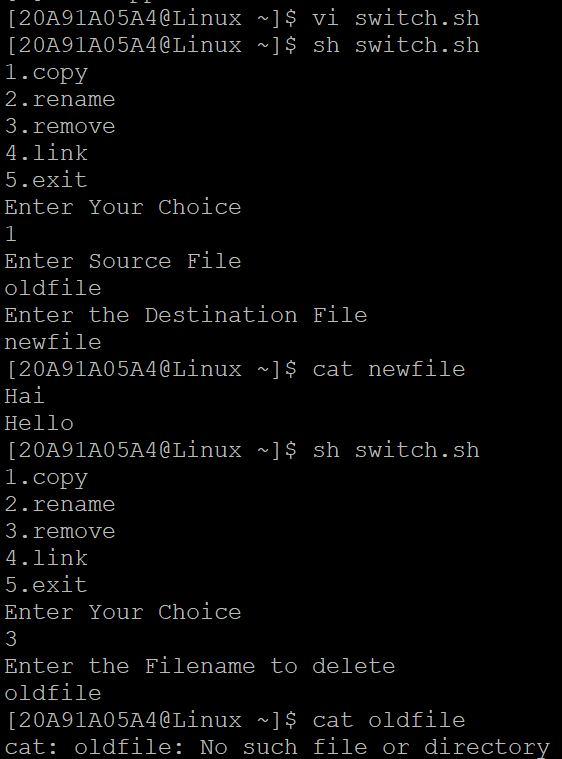
;;

5)exit 0

;;

Esac

**OUTPUT:**



**9a**)Write a Shell Script that takes a login name as command line Argument and reports when that person logs in

**AIM:** A Shell Script that takes a login name as command line Argument and reports when that person logs in

**PROGRAM:**

echo "Who are You?"

read user

echo $user

name=$(whoami)

if [ $user==$name ]

then

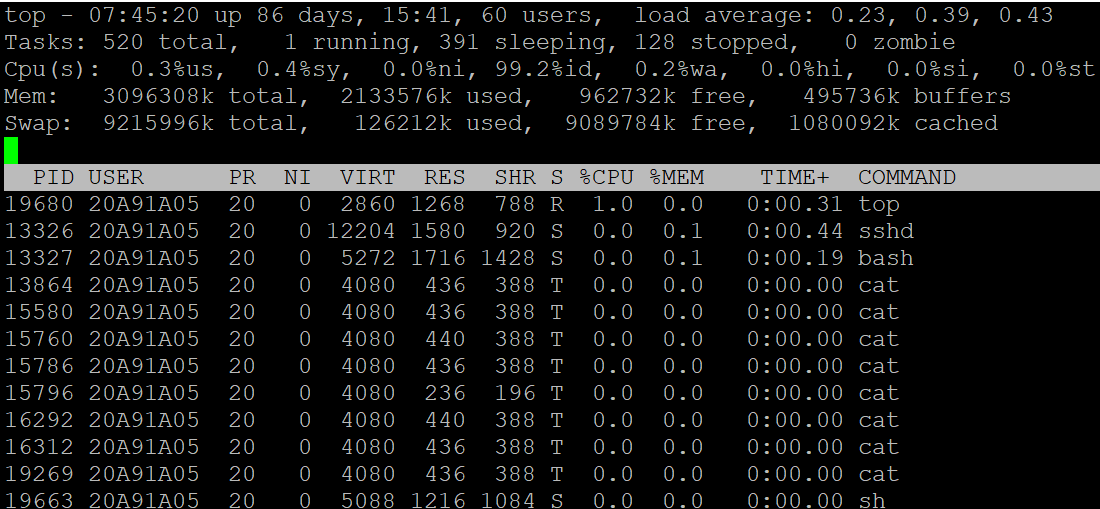
top -u $user

else

echo "Not logged in"

fi

**OUTPUT:**



**9b**) 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:** 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.

**PROGRAM:**

echo -n "Enter file1:"

read file1

echo -n "Enter File2:"

read file2

`cmp $file1 $file2>equal`

if [ ! -s equal ]

then

echo Same

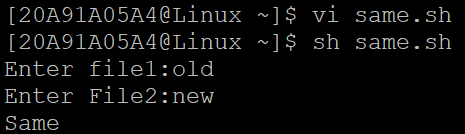
rm $file2

else

echo Different

fi

**OUTPUT:**

****

**10)** Write a C program that takes one or more file or directory names as a 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).

**PROGRAM:**

#include<stdio.h>

#include<unistd.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<fcntl.h>

void main()

{

int fd;

struct stat buf;

fd=open("hello.txt",O\_RDONLY|O\_CREAT,600);

if(fd!=-1)

{

if(fstat(fd,&buf)==0)

{

printf("Mode of File is %u",buf.st\_mode);

printf("\nSize of the File is %u",buf.st\_size);

printf("\nDevice Name %u",buf.st\_dev);

printf("\ninode of File is %u",buf.st\_ino);

printf("\nNo of Links are %u",buf.st\_nlink);

printf("\nOwner of a File is %u",buf.st\_uid);

printf("\nNo of Blocks is %u",buf.st\_blocks);

printf("\nGroup Owner is %u",buf.st\_gid);

printf("\nBlock Size of the File is %u",buf.st\_blksize);

printf("\nTime of Last Modified is %u",buf.st\_ctime);

}

else

printf("Error in fstat() syscall");

}

else

printf("Error in open() syscall");

}

**OUTPUT:**

