LAB MANUAL

Subject: UNIX Lab

Subject Code: SEITL403

Branch: IT

Class: S.E. I.T. (Div A & B)

Semester: IV

Academic Year: FH 2024

Subject In Charge

Prof. Pravin Patil
Prof.Vijaya Sagvekar

(HOD IT)

Dr. Pradip Mane

VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF
ENGINEERING AND VISUAL ARTS
Department of Information Technology

LIST OF EXPERIMENT

Sr.No	Name Of Experiment
1	To Study Unix and its installation
2	Study about basic commands of Unix.
3	Study about basic network commands.
4	Study of Unix editor
5	Study of Process Management Commands
6	Shell Programming.
7	Shell Programming.
8	Programming in shell scripts and sed.
9	Shell script with grep and awk commands.
10	Demonstrate Perl Script.

Prof. Pravin Patil
Prof.Vijaya Sagvekar

Dr. Pradip Mane
HOD IT

Ex.NO:1 **STUDY OF UNIX OS AND ITS INSTALLATION**

Aim:-To study about Unix OS and its Installation.

Theory

Introduction

An operating system is software that acts as an interface between the user and the computer hardware. It is considered as the brain of the computer. It controls and co-ordinates the internal activities of the computer and provides user interface.

The computer system is built with the following general components

i) Hardware
ii) Application Software
iii) Operating System

(i) **Hardware:** This includes the physical components such as CPU, Keyboard, Hard disk and Printer.

(ii) **Application Software:** These are the programs that are used to accomplish specific tasks.

(iii)**Operating System:** It is the component or the set of programs to manage and control the hardware as well as co-ordinate the applications. Each system must have at least had the hardware and the OS.

Services of an OS:

1. Process Management
2. File Management
3. I/O Management
4. Scheduling
5. Security Management

UNIX OPERARTING SYSTEM:

In the mid-1960s, AT &T Bell Laboratories developed a new OS called Multics. Multics was intended to supply large scale computing services as a utility; much like electrical power. In 1969 Ken Thompson, Dennis Ritchie and others developed and simulated an initial design for a file system that later evolved into the UNIX file system. The whole UNIX was rewritten in „C" language in 1973.Today, UNIX is a giant Os and is much powerful than most of its counter parts.

UNIX operating System is like a layer between the hardware and the applications that run on the computer .It has functions that run on the computer. It has functions that manage the executing applications. UNIX system is an OS, which includes the traditional system components. UNIX system includes a set of libraries and a set of applications.

KERNEL is the heart of UNIX OS that manages the hardware and the executing process. The UNIX system views each device as a file called a device file.

It implements security controls to protect the safety and privacy of information. The Unix System allocates resources including use of the CPU and mediates accesses to the hardware.

Application portability is the ability of a single application to be executed on various types of computer hardware without being modified. This is one of the important advantages of UNIX.

FEATURES OF UNIX:

1. **Multitasking**

   Multitasking is the capability of the Os to perform various tasks simultaneously. i.e. A single user can run multiple programs concurrently.

2. **Multiuser Capability**

   Multiuser capability allows several users to use the same computer to perform their

   tasks. Several terminals are connected to a single powerful computer and each user can work with their terminals.

3. **Security**

   Unix allows sharing of data. Every user must have a Login name and a password. So, accessing another user‟s data is impossible without permission.

4. **Portability**

   Unix is a portable because it is written in high level languages so it can run on different computers.

5. **Communication**

   Unix supports communication between different terminals connected to the Unix server and also between the users of one computer to the users of another computer located elsewhere in the network.

ORGANIZATION OF UNIX

The UNIX system is functionally organized at three levels and are:

1. The **kernel**, which schedules tasks and manages storage;
2. The **shell**, which connects and interprets users' commands, calls programs from memory, and executes them; and
3. The **tools and applications** that offer additional functionality to the OS

Kernel

The kernel is the heart of the system, a collection of programs written in C that directly communicate with the hardware.

Functions of Kernel

1. Allocating and deallocating memory to each and every process.
2. Receiving instructions from the shell and carrying them out.
3. Managing files that held on various storage devices.
4. Scheduling, Coordinating and assigning various input/output devices simultaneously. Providing Utility services.
5. Coordinating each and every process with signal handling.
6. Providing administrative functions or utilities.

Shell: It is the command interpreter of the OS. The commands given from the user are moved to the shell. The shell analyses and interprets these commands into the machine understandable form. The commands can be either typed in through the command line or contained in a file called shell script. Hence, Shell acts as an interface between the user and the kernel.

UNIX has a variety of shells, they are:

(i) **Bourne Shell:** It is developed by Steve Bourne and it is the most popular shell and widely used. This shell comes bundled with almost every Unix system.

(ii) **Korn shell:** It is developed by David G.Korn. This is superset of Bourne shell and it has more capabilities.

(iii) **C Shell:** It is developed by Bil Joy. It is similar to C Programming language.

1. All communications between user and Kernel takes place through the shell.
2. It allows the tasks to run on background.
3. It also enables us to construct scripts like a programming language.
4. A group of files can be executed using a single command.

Starting a UNIX session – Logging In

A user of Unix based system works as a user terminal. After the boot procedure is completed, that is the operating system is loaded in memory, the following message appears at each user terminal:

Logging

Each user has a identification called the user name are the login name which has to be entered when the login: message appears. The user is then asked to enter the password. Unix keeps track of all the Unix user names and the information about identity in a special file. If the login name entered does not match with any of the user names it displays the login message again. This ensures that, only authorized people use the system. When a valid user name is entered at the terminal the dollar symbol is displayed on the screen this is the Unix prompt.

Ending a UNIX session –Logging Out

By pressing the ctrl and „d keys" together or typing exit at the dollar prompt

Installation

As the open source revolution grows around the world, more and more people are starting to switch over to the Linux Operating System and pre-eminent of all the Linux OS is the Red Hat Linux, owned and distributed by the Red Hat Inc. However, installation of Linux itself is seen as a rather arduous and herculean task among many beginners/inexperienced users. As a result this document has been formulated as a step-by-step guide to ensure that everybody can install and use Red Hat Linux seamlessly and with ease. So lets start the installation, here are the steps to easily install red hat linux :

Step 1 – Insert the Red Hat Linux DVD into the DVD-drive of your computer. As soon as the following screen pops up, press 'Enter' to install Red Hat Enterprise Linux (RHEL) through GUI mode.

Step 2– RHEL installer would then prompt you conduct a check as to whether the CD media from which you're installing is functioning correctly or not. Choose 'Skip', press enter and the installation would begin.

Step 3– Next, we need to select the language- English or any other language as per your preference, and then press 'Next' .

Step 4– In this step, the RHEL installer would ask you about the appropriate type of keyboard for the system. We take the 'US English' keyboard, you can pick any other option depending on the type of your keyboard. Then press 'Next' to move to the next step.

Step 5– Next, the installer would ask for an 'installation number' if you wish to install full set of Red Hat functionalities. Enter the installation number and press 'OK' if you have an officially licensed installation number(for corporate clients that buy Red Hat's backup support and full features). Others can select 'Skip entering installation

number' and press 'OK' to proceed. RHEL would show a warning message, press 'Skip' in it to continue.

Step 6– The Red Hat installer would then require you to create partitions in your computer's hard disk for the installation. You can do it in four ways but the simplest way is to select 'Use free space on selected drives and create default layout' as this option will not affect any other OS residing in your system.

Check the 'review and modify portioning layout' to create partitions and click next.

Step 7– In this step you must create the required system partitions and mount points such as '/boot', '/home', 'swap' etc which are required for the Linux's proper functioning.

To create different partitions such as /home, /varetc, click on 'New' to create the partitions.

Then, select /home in the mount point and choose 'ext3' as the file system and give the desired size for it and then click 'OK'. Similarly also create /boot and /var. Also, create a swap partition by clicking on 'New' and then choosing the filesystem as 'swap' and also give the size of Swap partition.(Usually size of swap partition SHOULD BE twice the size of RAM available to the system but you can keep its size less than that too). Once you have made all the desired partitions and given their mount points, click 'Next' to continue installation.

Step 8– This step pertains to the default OS that will be loaded by the GRUB loader

(Note- If you have multiple Operating Systems installed, you would see multiple options here and you have to check in front of the OS name that you want to be loaded by default when the system is started.) Click 'Next' to continue.

Step 9– This step pertains to the network settings of the Linux system that you are going to install. You can select the Ethernet devices through which the system would communicate with other devices in the network. You can also provide the hostname, Gateway address and DNS address to the system during this step. (However it's better to adjust these settings once the system has been fully installed).

Step 10– The next step is to adjust the system clock to your particular time zone. Select your time zone and then click 'Next'.

Step 11 – This is a very important step that deals with the root(super-user) password for the system . Type the password and confirm it and then click next.

Step 12 – The RHEL installer would then prompt you about if you wish to install some extra 'Software Development' or 'Web Server' features. By default, keep it at 'Customize later' and press 'Next'.

Step 13– This next step will initiate the installation of Red Hat Linux, press 'Next' to begin the process.

Step 14– Upon the completion of installation you should the following screen. Press Reboot and you'd be ready to use your newly installed Red Hat Linux OS.

Conclusion:-Hence we studied Unix OS and its installation successfully.

Ex. No: 2 BASIC UNIX COMMANDS

Aim:

To study the logic basic UNIX command.

Login:

Click „start‟ button in the desktop. Then click on the, Run‟ command and then enter „telnet‟.Then give the value of the port address.

Start ⟶ Run ⟶ Telnet{port address}
After getting connected login:{user name}

Password:*********

Logout:

To logout from UNIX type ‟exit/ctrl+d‟

GENERAL PURPOSE COMMANDS:

1) **Command:** date

Purpose – used to display the date and time
Syntax: $ date

Example $date
Thu Dec 25 11:14:59 UTC 2013

Format	Purpose	Example	Result
+ % m	To display only month	$ date + % m	6
+ % h	To display month name	$ date + % h	June
+ % d	To display day of month	$ date + % d	01
+ % y	To display last two digits of the year	$ date + % y	9
+ % H	To display hours	$ date + % H	10
+ % M	To display minutes	$ date + % M	45
+ % S	To display seconds	$ date + % S	55

2) **COMMAND** : cal
PURPOSE– This command is used to display the month or year calendar. **SYNTAX**: $ cal 2 2013

3) **COMMAND**: echo
PURPOSE : This command is used to display the given text.
SYNTAX : $echo<text>
EXAMPLE : echo “Hello world”
Hello world

4) **COMMAND : lp**
PURPOSE – used to take printouts **SYNTAX**: $ lp filename

Cancel — Cancels a print job under the System operating system.
lpq — List the status of available printers.
lpr — Submit print requests.
lprm — Remove requests from the print queue.
lp — Print a file on the System operating system.

5) **COMMAND** : who am i
PURPOSE : This command is used to display the current working user.
SYNTAX EXAMPLE
: $who am i
: $who am i
User1 tty1 sep 04 09:30

7) **COMMAND** : clear
PURPOSE : This command is used to clear the screen.
SYNTAX: $clear

8) **COMMAND** : time
PURPOSE : This command is used to figure out the time taken to execute the Command.
SYNTAX : $time command
EXAMPLE : $time cat

9) **COMMAND**: man
PURPOSE : used to provide manual help on every UNIX commands.
SYNTAX : $man
EXAMPLE :$ man pwd
NAME
Pwd-print name of current/working directory
SYNOPSIS
Pwd [OPTION]

10) **COMMAND :** uptime
PURPOSE: tells you how long the computer has been running since its last reboot or power-off.
SYNTAX: $ uptime

11) **COMMAND :** uname
PURPOSE– it displays the system information such as hardware platform, system name and processor, OS type.
SYNTAX: $ uname –a

12) **Command : cat**
Purpose – this create, view and concatenate files. **Creation**: Syn: $ cat > file
Viewing: Syn: $ cat filename

Add text to an existing file: Syn: $cat >> filename
Concatenate:
Syn: $ cat file1 file2 > file3
$ cat file1 file2 >> file3 (no overwriting of file3)
Example: $cat a3 Welcome to unix operating system

13) **COMMAND :** touch
PURPOSE: This command is used to create an empty file
SYNTAX: $touch filenames

14) **COMMAND: pwd** stands for Print Working Directory. It prints he path of the working directory, starting from the root.
Example: $pwd
/home/george

Conclusion: Thus we studied basic Unix commands

Ex. No: 3 STUDY OF NETWORKING COMMNADS

Aim: To study about networking commands.

Theory:

Hostname
hostname with no options displays the machine's hostname
hostname –d displays the domain name the machine belongs to
hostname –f displays the fully qualified host and domain name
hostname –i displays the IP address for the current machine
ping
It sends packets of information to the user-defined source. If the packets are received, the destination device sends packets back. Ping can be used for two purposes

1. To ensure that a network connection can be established.
2. Timing information as to the speed of the connection.

If you do ping www.yahoo.com it will display its IP address. Use ctrl+C to stop the test.

Ifconfig

ifconfig utility is used to configure network interface parameters. Mostly we use this command to check the IP address assigned to the system.

```
# ifconfig -a
```

traceroute

traceroute print the route packets take to network host. Destination host or IP is mandatory parameter to use this utility

```
[root@localhost ~]# traceroute geekflare.com
```

telnet:telnet connect destination host:port via a telnet protocol if connection establishes means connectivity between two hosts is working fine.

```
[root@localhost ~]# telnet geekflare.com 443
```

nmap is a very powerful command, which checks the opened port on the server.
Usage example:

```
nmap $server_name
```

Enable/Disable Network Interface

You can enable or disable the network interface by using ifup/ifdown commands with ethernet interface parameter.
#ifup eth0
#ifdown eth0

Conclusion: Thus we studied networking commands

Ex.No:4 STUDY OF UNIX EDITOR- VI

Aim: To study about Unix editor (Vi)

Theory

This section introduces the standard UNIX text editor, "vi". The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. vi has no menus but instead uses combinations of keystrokes in order to accomplish commands.

1. Starting the vi Editor:

There are following way you can start using vi editor:

Command	Description
vi Filename	Creates a new file if it already does not exist, otherwise opens existing file.
vi -R filename	Opens an existing file in read only mode.
view filename	Opens an existing file in read only mode.

2. Operation Modes:

While working with vi editor, it comes across the following two modes:

i) **Command mode:** This mode enables you to perform administrative tasks such as saving files, executing commands, moving the cursor, cutting (yanking) and pasting lines or words, and finding and replacing. In this mode, whatever you type is interpreted as a command.

ii) **Insert mode:** This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and finally it is put in the file .

The vi always starts in command mode. To enter text, you must be in insert mode. To come in insert mode you simply type **i**. To get out of insert mode, press the **Esc** key, which will put you back into command mode.

3. Closing and Saving Files

When editing a file in vi, it is actually editing a copy of the file rather than the original. The following describes the methods to be used when closing a file, quitting vi, or both.

i) Quitting and Saving a File

The command **ZZ** (notice that it is in uppercase) will allow you to quit vi and save the edits made to a file. You will then return to a Unix prompt. Note that you can also use the following commands:

:w
:q
:wq

ii) Quitting without Saving Edits

Sometimes, when you create a mess (when you first start using vi this is easy to do!) you may wish to erase all edits made to the file and either start over or quit. To do this, you can choose from the following two commands:

:e! reads the original file back in so that you can start over.
:q! wipes out all edits and allows you to exit from vi.

4. Steps for Inserting Text With "vi"
The following shows how to open a file, enter text, and then save the changes to the file.

Step 1: At the command line, type "vi doc1".
Step 2: Type the letter "i". This puts you into insert mode.
Step 3: Enter the following text into your document:
This is my first vi document.
This is the second line of the document.
Step :4 Hit the "<Esc>" key. This takes you back into command mode.
Step 5: Now enter ":wq" to save the changes to the file. This stands for "write and quit". It returns to the UNIX command line.

5. Cursor Movement

It must be in command mode if we wish to move the cursor to another position in the file. If we have just finished typing text, we are still in insert mode and will need to press **ESC** to return to the command mode.

i) Moving One Character at a Time

In case of direction keys to move up, down, left and right in the file, the following keys move in the following directions:

`h` left, one space. `K` up, one line.
`J` down, one line. `l` right, one space

ii) Moving among Words and Lines

While these four keys (or your direction keys) can move anywhere to go in the file, there are some shortcut keys that can be used to move a little more quickly through a document. To move more quickly among words, use the following:

moves the cursor forward one word
moves the cursor backward one word (if in the middle of a word, b will move you to the beginning of the current word).
moves to the end of a word.

iii) Screen Movement

To move the cursor to a line within your current screen use the following keys:
moves the cursor to the top line of the screen.
Mmoves the cursor to the middle line of the screen.
Lmoves the cursor to the last line of the screen.

To scroll through the file and see other screens use:

ctrl-f scrolls down one screen
ctrl-b scrolls up one screen

ctrl-u scrolls up a half a screen
ctrl-d scrolls down a half a screen

6. Basic Editing

To issue editing commands, it must be in command mode. As mentioned before, commands will be interpreted differently depending upon whether they are issued in lower or upper case. Also, many of the editing commands can be preceded by a number to indicate a repetition of the command.

i) Deleting (or Cutting) Characters, Words, and Lines

To delete a character, first place your cursor on that character. Then, you may use any of the following commands:

deletes the character under the cursor.
deletes the character to the left of your cursor.
dw deletes from the character selected to the end of the word.
deletes all the current line.
deletes from the current character to the end of the line.

ii) Pasting Text using Put

Often, when we delete or cut text, we wish to reinsert it in another location of the document. The Put command will paste in the last portion of text that was deleted since deleted text is stored in a buffer. To use this command, place the cursor where the deleted text should appear. Then use **p** to reinsert the text. If we are inserting a line or paragraph use the lower case **p** to insert on the line below the cursor or upper case **P** to place in on the line above the cursor.

iii) Copying Text with Yank

To make a duplicate copy of existing text, use the yank and put commands to accomplish this function. Yank copies the selected text into a buffer and holds it until another yank or deletion occurs. Yank is usually used in combination with a word or line object such as the ones shown below:

yw copies a word into a buffer (7yw copies 7 words)
copies a line into a buffer (3yy will copy 3 lines)

Once the desired text is yanked, place the cursor in the spot in which you wish to insert the text and then use the put command (**p** for line below or **P** for line above) to insert the contents of the buffer.

iv) Replacing or Changing Characters, Words, and Lines

When using the following commands to replace text, put temporarily into insert mode so that we can change a character, word, line, or paragraph of text.

replaces the current character with the next character we enter/type. Once we enter the character we are returned to command mode.

puts in overtype mode until we hit **ESC** which will then return to command mode.

Cw changes and replaces the current word with text that we type. A dollar sign marks the end of the text we are changing. Pressing **ESC** when you finish will return you to command mode.

v) Undoing

When making a mistake we can undo it. **DO NOT** move the cursor from the line where you made the change. Then try using one of the following two commands:

undoes the last change you made anywhere in the file. Using **u** again will "undo the
undo".

undoes all recent changes to the current line. You can not have moved from the line to recover the original line.

Here are the key points to success with vi:

i) Must be in command mode to use commands. (Press Esc twice at any time to ensure that you are in command mode.)

ii) Must be careful to use the proper case (capitalization) for all commands.
iii) Must be in insert mode to enter text.

Conclusion:-Hence we studied VI editor.

Ex.No:5 STUDY OF PROCESS MANAGEMENT COMMANDS

Aim: To study Job & Process Management Commands.

Theory:

1) **exit** – terminates a process **syn:** $ exit
2) **kill** – terminates or send a signal to process **syn:** $ kill
3) **passwd** – create or change a password **syn:** $ passwd
4) **telnet** – connect to remote machine using the telnet protocol **syn:** $ telnet
5) **COMMAND :** ps
   PURPOSE : Each process is given an unique identification number. The second column display the name of the terminal from which the process is being controlled.

EXAMPLE FOR PROCESS MANAGEMENT COMMANDS

[cp2@localhost cp2]$ **ps**

```
  PID TTY      TIME CMD
8649 pts/2    00:00:00 bash
12498 pts/2   00:00:00 ps
```

6) **top**
   The top command is probably the most well know utility for displaying the most resource intensive processes on the system. For the most part you can get away with just running top and looking at the output. # top

Conclusion: Thus we studied process management commands

EX.NO. 6 SHELL PROGRAMMING

Ex.No:6a SHELL SCRIPT TO DISPLAY HELLO WORLD

AIM

To write a shell script to find display hello world.

```
# This is comment line
echo "Hello World"
ls
date
```

Ex.No:6b SHELL SCRIPT TO DEVELOP SCIENTIFIC CALCULATOR

AIM

To write a shell script to develop scientific calculator.

```
clear
sum=0
i="y"

echo " Enter one no."
read n1
echo "Enter second no."
read n2
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
```

```
case $ch in
    1)sum=`expr $n1 + $n2`
     echo "Sum ="$sum;;
        2)sum=`expr $n1 - $n2`
     echo "Sub = "$sum;;
    3)sum=`expr $n1 \* $n2`
     echo "Mul = "$sum;;
    4)sum=`expr $n1 / $n2`
     echo "Div = "$sum;;
    *)echo "Invalid choice";;
esac
echo "Do u want to continue ?"
read i
if [ $i != "y" ]
then
    exit
fi
done
```

OUTPUT

****

```
$ sh calculator.sh
Enter any no.
121
 Enter one no.
 21
 Enter second no.
 58
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
 Enter your choice
 1
 Sum =79
 Do u want to continue ?
 y
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
 Enter your choice
 2
 Sub = -37
 Do u want to continue ?
 y
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
```

```
Enter your choice
3
Mul = 1218
Do u want to continue ?
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
4
Div = 0
Do u want to continue ?
n
```

Ex.No:6c FIND THE GIVEN NUMBER IS EVEN OR ODD

AIM: To write a shell script to find whether the given number even or odda

```
echo -n “enter any integer number to find even and odd :-”
read number
rem=`expr $number % 2`
if test $rem -eq 0
then
echo “number is even”
else
echo “number is odd”
fi
```

Ex.No:6d PRIME OR NOT

AIM

To write a shell script to find whether the given number is prime or not

ALGORITHM

1.Start the program.

2. read n and initialize i=2
   3.Repeat the step 4 upto n

4.Check whether n%i==0

5. If the value is true display it as prime else display as not prime.
6.Stop the program.

PROGRAM

```
echo –n “enter the number”
```

```
read n
i=2
while [ $i -le $n ]do
if [ `expr $n % $i` -eq 0 ]
then
break
fi
i=`expr $i + 1`
done
if [ $n -eq $i ]
then
echo "Prime Number"
else
echo "Not Prime"
fi
```

OUTPUT

```
$ sh prime.sh

Enter the number
17
Prime number
```

Ex.No:6e PALINDROME OR NOT

AIM

To write a shell script to find whether the given number palindrome or not

ALGORITHM

1. Start the program.
2. Read n and assign it to t.
3. Initialize d=0.

4.Calculate the palindrome using,

r=n%10
d=d*10+r
n=n/10

4.Repeat step3 until n>0

5. Check whether the number and reversed number, when it is equal then display the number is palindrome. Else display it is not a palindrome

PROGRAM

```
echo –n “enter the number”
read n
t=$n
d=0
while [ $n -gt 0 ]
do
r=`expr $n % 10`
```

```
d=`expr $d \* 10 + $r`
n=`expr $n / 10`
done
if [ $d -eq $t ]
then
echo “Number is palindrome”
else
echo “Number is not palindrome”
fi
echo
```

OUTPUT

```
 $ sh palindrome.sh

Enter a number
121
Number is palindrome
```

Ex.No:6f **Binary search**

Aim: Write a shell program to search for a given number from the list of numbers provided using binary search method

```
Echo “Enter array limit”
Read limit
Echo “Enter elements”
n = 1
while [ $n -le $limit ]
do
Read num
eval arr$n = $num
n = `expr $n + 1`
done
Echo “Enter key element”
Read key
low = 1
high = $n
found = 0
while [ $found -eq 0 -a $high -gt $low ]
do
mid = `expr \( $low + $high \) / 2`
eval t = \$arr$mid
if [ $key -eq $t ]
```

```
then
found = 1
elif [ $key -lt $t ]
then
high = `expr $mid – 1`
else
low = `expr $mid + 1`
fi
done
if [ $found -eq 0 ]
then
Echo “Unsuccessful search”
else
Echo “Successful search”
fi
```

Conclusion: Thus the shell programs are executed successfully

EX.NO: 7 SHELL PROGRAMMING

Ex.No.7a Generate even numbers and calculate its sum

AIM

To write a shell script to generate even numbers and calculate its sum.

ALGORITHM

1.Read the value of n.
2.Assume i=2,sum=0 and j=0
3.Find the even numbers j=j+1
4.Print the value of sum=sum+i
5.Print the value of sum.

PROGRAM

```
echo –n “Enter the limit”
read n
i=2
sum=0
j=0
while [ $i  –le $n ]
do
echo -n “$i”
sum=` expr $sum + $i `
```

```
j=`expr $j + 1`
i=`expr $i + 2`
done
echo -n “the sum of the first $j even numbers is $sum ”
```

OUTPUT

```
$ sh sum.sh
Enter the limit
4
the sum of the first 4 even numbers is 20
```

RESULT

Thus the shell program to generate even numbers and calculate its sum is executed and output is verified successfully.

Ex.No:7b FIBONACCI SERIES

AIM

To write a shell program to generate Fibonacci

ALGORITHM

1.Start the program
2.Read the limit
3.Set first term to zero i.e (b=0)

4.Set second term to 1i.e (c=1)
5.Check the limit is greater than 2 if its reduce the limit by 2 or print the first two terms
6.To find the next term add the previous two terms of the series and previous it and print it and repeat the steps with the limit.
7.Stop the program

PROGRAM

```
echo –n “Enter the limit”
read n
echo “The Fibonacci series…:”
b=0
c=1
d=0
i=0
if [ $n –ge 2 ]
then
echo –n “$b”
echo –n “$c”
n=` expr $n – 2 `
while [ $i –lt $n ]
do
a=` expr $b + $c `
b= $c
c=$a
echo –n “$c”
```

```
i=` expr $i + 1`
done
else
echo –n “$b”
fi
```

OUTPUT

```
 $ sh fibonacci.sh
Enter the limit
10
The Fibonacci series…
0
1
1
2
3
5
8
13
21
34
```

RESULT

Thus the shell program to generate Fibonacci series is executed and output is verified successfully.

.

Ex.No:7c COPY A STRING INTO ANOTHER STRING

AIM:

To write a shell script to copy a string into another string, character by character.

ALGORITHM

1. Start the program
2. Read the string from the user
3. Find the length and the string using wc command
4. Set the initialization value of the while loop to 1 ie a=1
5. Using cut command extract each character according to initialization value and store it in another variable
6. Repeat increment the initialization value by 1 ie a=a+1
7. Repeat the steps 4,5 until the lengths of the string.
8. Print the copied string
9. Stop the program

PROGRAM

```
echo-n”enter the string:”
read str
```

```
len=`echo $str | wc -c`
a=1
while [ $a –le $len ]
do
rev=`echo $str | cut -c $a`
a=`expr $a + 1`
temp=` echo $temp $rev`
done
echo “the copied string is $temp”
```

OUTPUT

```
$ sh stringcopy.sh
Enter the string
Hello
The copied string is Hello
```

RESULT

Thus the shell program to copy the string is executed and output is verified successfully.

Ex.No:7d REVERSE THE STRING

AIM

To write a shell script to reverse the string

ALGORITHM

1. Start the program
2. Read a string
3. Find the length of the string by using w c command
4. Extract the right most character in a string by using cut command and store that character in temp variable
5. Extracting each character is stored one after one in rev. variable
6. Decrement the length of the string
7. Repeat the steps 4,5,6 upto the length of the string is greater than zero
8. Print the reverse string i.e rev.variable
9. Stop the program

PROGRAM

```
echo-n”enter the string”
read str
len=`echo $str | wc -c`
while [ $len –gt 0 ]
do
temp=`echo $str | cut –c $len`
rev=`echo $rev $temp`
len=`expr $len – 1`
done
echo” the reversed string is $rev”
```

OUTPUT

```
$ sh strrev.sh
Enter the string
Welcome
The reversed string is emoclew
```

RESULT

Thus the shell program to reverse a string is executed and output is verified successfully.

Ex.No:7e CONVERSION OF LOWER CASE LETTERS TO UPPER CASE

AIM

To write a shell script to convert lower case letter to its upper case

ALGORITHM

1. Start the program
2. Read the string from users
3. Assign temp=str/tr[a – z][A-Z]
4. Print the case changed string
5. Stop the program

PROGRAM

```
echo-n"enter a string:"
read str
temp=`echo $str | tr[a-z] [A-Z]`
echo "the case changed string is $temp"
```

OUTPUT

```
$ sh case.sh
enter a string:
lotus
the case changed string is LOTUS
```

RESULT

Thus the shell program to the conversion of lowercase letters to uppercase is executed and output is verified successfully.

EX: NO 8 STUDY OF SHELL SCRIPTS AND SED

Aim: - Use sed instruction to process /etc/passwd file.

sed - stream editor for filtering and transforming text

SYNTAX

```
sed [OPTION]... {Script-only-if-no-other-script} [Input-file]...
```

Example:

```
echo "line addressing"
```

sed -n '1,2p' /etc/passwd
sed -n '5,6p
7,9p
$p' /etc/passwd
echo “Context Addressing”
sed -n '/jnec/p' /etc/passwd

DESCRIPTION
Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.
-n, --quiet, --silent suppress automatic printing of pattern space
-e script, --expression=script add the script to the commands to be executed
-f script-file, --file=script-file add the contents of script-file to the commands to be execut ed
-i[SUFFIX], --in-place[=SUFFIX] edit files in place (makes backup if extension supplied)
-c, --copy use copy instead of rename when shuffling files in -i mode (avoids change of input file
ownership)
-l N, --line-length=N specify the desired line-wrap length for the 'l' command
--posix disable all GNU extensions.
-r, --regexp-extended use extended regular expressions in the script.
-s, --separate consider files as separate rather than as a single continuous long stream.
#sed 'ADDRESSs/REGEXP/REPLACEMENT/FLAGS' filename
#sed 'PATTERNs/REGEXP/REPLACEMENT/FLAGS' filename
s is substitute command
/ is a delimiter
REGEXP is regular expression to match, REPLACEMENT is a value to replace, and FLA
GS can be any of the following
g: Replace all the instance of REGEXP with REPLACEMENT
n : Could be any number, replace nth instance of the REGEXP with REPLACEMENT.
p : If substitution was made, then prints the new pattern space.
i: match REGEXP in a case-insensitive manner. w file If substitution was made, write out the result to the given file.
We can use different delimiters (one of @ %; :) instead of /
OUTPUT:
Line Addressing
1st and 2nd line /etc/passwd file will be printed.5th and 6th line, 7 to 9 lines and last line of /etc/passwd file will be printed. Context Addressing whenever string 'jnec' matches i n /etc/passwd that line will be printed.
CONCLUSIONS:
With the help of given procedure and information about the commands we can process sed
Instructions on /etc/passwd file.

Conclusion: Thus we studied shell programming with sed.

EX: NO:9 STUDY OF GREP AND AWK SCRIPTS

Grep:

---

grep is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type % grep science science.txt As you can see, grep has printed out each line containg the word science

.

Or has it ????

Try typing

% grep Science science.txt

The

grep

command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the

-

i option, i.e. type

% grep

-

i science science.txt

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

% grep

-

i 'spinning top' science.txt

Some of the other options of grep are:

-

v

display th

ose lines that do NOT match

-

n

precede each matching line with the line number

-

c

print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time. For example, the numbe

r of lines without the words science or

Science is

% grep

-

ivc science science.txt

EX.NO 9a

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

```
echo “Enter a word”
read word
echo “Enter the
filename”
read file
nol=grep
-
c $word $file
echo “ $nol times $word present in the $file”
```

Ex.No.9b.

AIM: Develop an interactive awk script that asks for a word and a file name and then tells how many lines contain that word.

```
echo “Enter a word”
read word
echo “Enter the filename”
read file
nol=awk „/$word/ { print NR }‟ Infile
echo “ $nol times $word pre
sent in the $file”
```

Conclusion:

Thus we studied grep and awk script.

EX.NO.10 STUDY OF PERL SCRIPTS

Aim: -

Write a perl program to demonstrate the use of array variables.

STANDARD PROCEDURE:
• Step 1: start UNIX OS in your computer and login in it and enter Username and Password.
• Step 2: Create a folder with your Id Number or Name Followed by Roll No.
• Step 3: now go to your folder from the terminal and after that open the VI editor with the desired program name with extension . pl
• Step 4: now write your program and quit back to the terminal.
• Step 5: After writing program make it executable by using
$chmod u+ x program name.
• Step 6: Now run the program using any one of the method...
• perl program name.pl

THEORY:
Array Variables:
An array in perl program is a list of scalars (numbers and strings).Array variables have the same format as scalar variables except that they are prefixed by an @ symbol.The statement
@food = (“Apples”,”pears”,”Banana”);
@music =(“whistle”,”flute”);
assigns a three element list to the array variable @food and a two element list to the array variable @music. The array is accessed by using indices starting from 0, and square brackets are used to specify the index.The expression $food[2] returns Banana and @ changed to $ because banana is a scalar. We can also add an element to an array later by using push statement. push (@food ,”Value”); To remove any element from an array we can use the pop statement. pop (@food) removes last element from an array.
Control Structures Perl supports lots of different kinds of control structures which tend to be like those in 'C' Programming but are very similar to Pascal, too. Here we discuss a different one. Foreach To go through each line of an array or other list-like structure Perl uses the foreach structure. This has the form

```
foreach $variable (@arr_name) #visit each element in turn
{
print “$variable ” #Print the variable
print “Message”
}
```

Example:

```
@food = (“Apples”,”Pears”,”Banana”);
for each $item (@food) #visit each element in turn and call it item
{
print “$item \n” #Print the item
print “Very Good! \n” #That was nice
}
```

OUTPUT:

```
Apples
Very Good!
Pears
Very Good!
Banana
Very Good!
```

Conclusion:

Thus we studied Perl scripts.