**Practical-1**

**Aim: To study…**

**1) UNIX**

**2) Difference between LINUX and UNIX**

**3) UNIX system organization**

**4) Types of shell**

**UNIX**

**What is UNIX?**

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.

## Types of UNIX:

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.

Here in the School, we use Solaris on our servers and workstations, and Fedora Linux on the servers and desktop PCs.

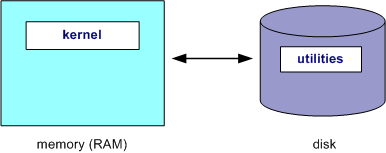
**(2) Difference between LINUX and UNIX:**

|  |  |  |
| --- | --- | --- |
| Comparison | Linux | Unix |
| Definition | It is an open-source operating system which is *freely available to everyone*. | It is an operating system which *can be only used by its copyrighters*. |
| Examples | It has different distros like Ubuntu, Redhat, Fedora, etc | IBM AIX, HP-UX and Sun Solaris. |
| Users | Nowadays, Linux is in great demand. Anyone can use Linux whether a home user, developer or a student. | It was developed mainly for servers, workstations and mainframes. |
| Usage | Linux is used everywhere from servers, PC, smartphones, tablets to mainframes and supercomputers. | It is used in servers, workstations and PCs. |
| Cost | Linux is freely distributed,downloaded, and distributed through magazines also. And priced distros of Linux are also cheaper than Windows. | Unix copyright vendors decide different costs for their respective Unix Operating systems. |
| Development | As it is open source, it is developed by sharing and collaboration of codes by world-wide developers. | Unix was developed by AT&T Labs, various commercial vendors and non-profit organizations. |
| Manufacturer | Linux kernel is developed by the community of developers from different parts of the world. Although the father of Linux, Linus Torvalds oversees things. | Unix has three distributions IBM AIX, HP-UX and Sun Solaris. Apple also uses Unix to make OSX operating system. |
| GUI | Linux is command based but some distros provide GUI based Linux. Gnome and KDE are mostly used GUI. | Initially it was command based OS, but later Common Desktop Environment was created. Most Unix distributions use Gnome. |
| Interface | The default interface is BASH (Bourne Again SHell). But some distros have developed their own interfaces. | It originally used Bourne shell. But is also compatible with other GUIs. |
| File system support | Linux supports more file system than Unix. | It also supports file system but lesser than Linux. |
| Coding | Linux is a Unix clone,behaves like Unix but doesn't contain its code. | Unix contain a completely different coding developed by AT&T Labs. |
| Operating system | Linux is just the kernel. | Unix is a complete package of Operating system. |
| Security | It provides higher security. Linux has about 60-100 viruses listed till date. | Unix is also highly secured. It has about 85-120 viruses listed till date |

**(3) UNIX system organization:**

As with most modern operating systems, the Unix OS is also madeup of many different components. In a very general sense, Unix is divided into two main components, the kernel component and the utilities. The kernel, which is critical to the operation of the OS, is loaded into Random Access Memory (RAM) by the boot loader, where it remains memory resident for as long as the machine remains powered on. The utilities are programs which (typically) reside on a disk device (e.g. a harddrive). Individual utillities are loaded into RAM as needed or requested and are discarded from RAM upon completion.

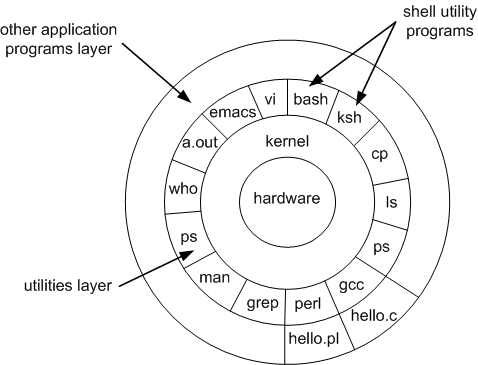
* The relationship between the kernel and the utilities is pictured below.



Perhaps the most "important" or well known of the Unix utilitites is known as the Unix **s**hell. The shell is the mechanism which allows users to enter commands to run other utility programs. There are several popular Unix shell programs which will be discussed later. What is important to keep in mind that the shell is merely another Unix utility program, which is typically loaded at login.

The diagram below provides another visual representation of the organization of the Unix OS. At the core of the OS is the hardware, which is managed by the surrounding outer layer, the kernel. In the next outer layer come the utilities. Many of the utilities are system commands, but these can also be user written programs as shown by the a.out program.

* Finally in the outermost layer are other application programs which can be built on top of lower layer programs.



* As RAM has become plentiful and affordable, many of the frequently used utility programs have become "built-in" to the shell.
* The program a.out is the generic name given to executable Images produced by the C compiler.

**(4) Types of shell:**

### **1. The Bourne Shell (sh)**

Developed at AT&T Bell Labs by Steve Bourne, the Bourne shell is regarded as the first UNIX shell ever. It is denoted as sh. It gained popularity due to its compact nature and high speeds of operation.

This is what made it the default shell for Solaris OS. It is also used as the default shell for all Solaris system administration scripts. Start reading about shell scripting here.

**However, the Bourne shell has some major drawbacks.**

* It doesn’t have in-built functionality to handle logical and arithmetic operations.
* Also, unlike most different types of shells in Linux, the Bourne shell cannot recall previously used commands.
* It also lacks comprehensive features to offer a proper interactive use.

The complete path-name for the Bourne shell is /bin/sh and /sbin/sh. By default, it uses the prompt # for the root user and $ for the non-root users.

### **2. The GNU Bourne-Again Shell (bash)**

More popularly known as the Bash shell, the GNU Bourne-Again shell was designed to be compatible with the Bourne shell. It incorporates useful features from different types of shells in Linux such as Korn shell and C shell.

It allows us to automatically recall previously used commands and edit them with help of arrow keys, unlike the Bourne shell.

The complete path-name for the GNU Bourne-Again shell is /bin/bash. By default, it uses the prompt bash-VersionNumber# for the root user and bash-VersionNumber$ for the non-root users.

### **3. The C Shell (csh)**

The C shell was created at the University of California by Bill Joy. It is denoted as csh. It was developed to include useful programming features like in-built support for arithmetic operations and a syntax similar to the C programming language.

Further, it incorporated command history which was missing in different types of shells in Linux like the Bourne shell. Another prominent feature of a C shell is “aliases”.

The complete path-name for the C shell is /bin/csh. By default, it uses the prompt hostname# for the root user and hostname% for the non-root users.

### **4. The Korn Shell (ksh)**

The Korn shell was developed at AT&T Bell Labs by David Korn, to improve the Bourne shell. It is denoted as ksh. The Korn shell is essentially a superset of the Bourne shell.

Besides supporting everything that would be supported by the Bourne shell, it provides users with new functionalities. It allows in-built support for arithmetic operations while offereing interactive features which are similar to the C shell.

The Korn shell runs scripts made for the Bourne shell, while offering string, array and function manipulation similar to the C programming language. It also supports scripts which were written for the C shell. Further, it is faster than most different types of shells in Linux, including the C shell.

The complete path-name for the Korn shell is /bin/ksh. By default, it uses the prompt # for the root user and $ for the non-root users.

### **5. The Z shell (zsh)**

The Z Shell or zsh is a sh shell extension with tons of improvements for customization. If you want a modern shell that has all the features a much more, the zsh shell is what you’re looking for.

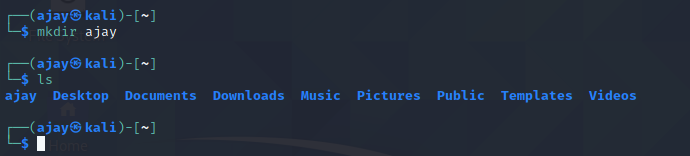
**Practical-2**

**Aim :- Study of Basic commands of Linux/UNIX.**

1. **mkdir**

Syntax: mkdir [option] …. [directory]

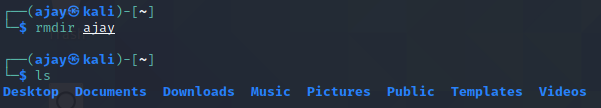
Description: Create the Directories, which are not already exit.

****

1. **rmdir**

Syntax: rmdir [option] ….. [directory]

Description: Remove directories if they are empty.

****

1. **ls**

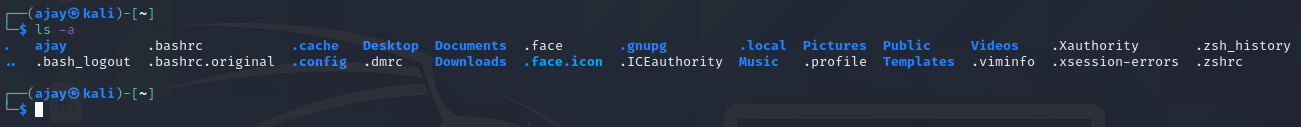
Syntax: ls [option] …. [file]

Description: List out all files in current directory.

**⇒** **ls -a**

Description: List out all files starting with . (dot)

**⇒** **ls –l**

Description: List files with long format and show permissions.

1. **Cd**

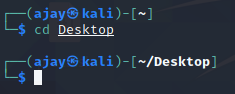
Syntax: cd [option] … [file]

**⇒ cd d1**

Description: Change directory to d1.

**⇒ cd . .**

Description: Move one directory back.



1. **Cat**

Syntax: cat [option] …. [file]

Description: Concatenate files to print on standard output.

**⇒ cat >[file]**

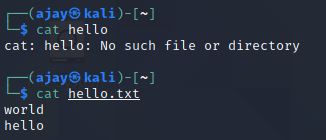
Description: Create file and write content in it.

**⇒ cat [file]**

Description: Show contents of file.

**⇒ cat [file1] [file2] >[file3]**

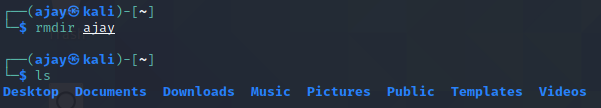
Description: Concate two files contents and create new file with this contents.

****

1. **Rm**

Syntax: rm [option] …. [file]

Description: Remove each specified files.

****

1. **Touch**

Syntax: touch [option] …. [file]

Description: Update the access and modification time of each file to current file.

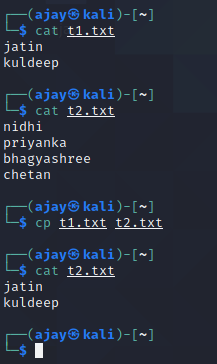
**⇒ touch -a [file]**

Description: Change only the access time.

1. **cp**

Syntax: cp [option] source destination/directory

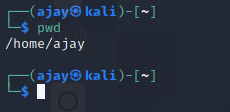
Description: It will copy the content of source file to destination file. If the destination file does not exist, it will be created. If it exists then it will be overwritten without any warning.

****

1. **pwd**

Syntax: pwd [option] ….

Description: Print the full name of current working directory.

****

1. **man**

Syntax: man command\_name [options]

Description: It is short form of manual. It provides in depth information about the requested command.



**Practical-3**

**Aim :- Study of Advance commands and filters of Linux/UNIX.**

1. **uname**

Syntax: uname [options]

Description: Used to print system Information.

-s prints the kernel name.

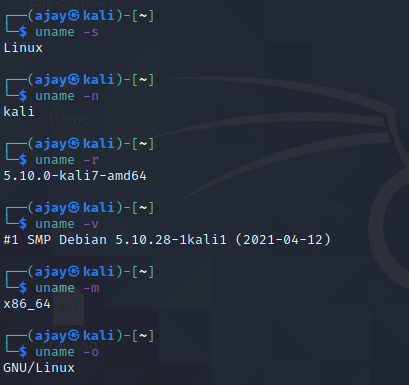
-n print the network node hostname.

-r prints the kernel release.

-v print the kernel version.

-m prints the machine hardware name.

-o prints the operating system.



1. **bc**

Syntax: bc [option]

Description: It is used for command line calculator. It is similar to basic calculator. bc starts by processing code from all the files listed on the command line in the order listed. After processing all the files bc reads from the standard input. Code is executed as it is read.

-q To avoid bc welcome message.

-l To include math library functionalities.

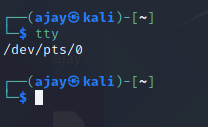
 **quit** to quit the calculator.

1. **tty**

Syntax: tty

Description: prints the file name of the terminal connected to standard input.

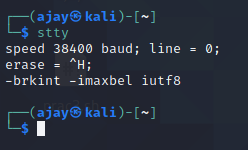
* It writes the name of the terminal that is connected to standard input into standard output.
* It does not require any additional argument.

****

1. **stty**

Syntax: stty

Description: It sets certain terminal I/O modes for the device that is the current standard input. Without arguments, it writes the settings of certain modes to standard output.

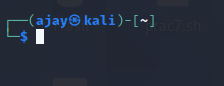


1. **clear**

Syntax: clear

Description: It clears the terminal screen.

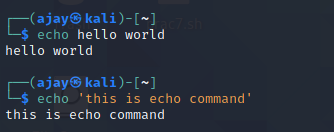
* It clears your screen if this is possible, including its scroll back buffer.
* Clear ignores any command-line parameters that may be present.



1. **echo**

Syntax: echo 'string'

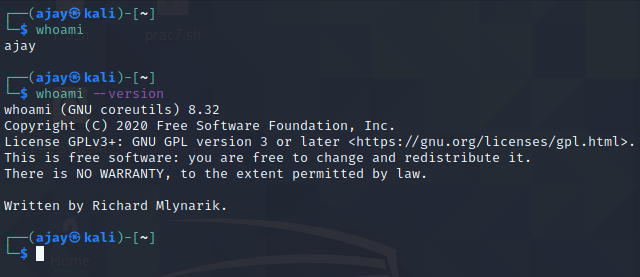
Description: It prints the given input string to standard output. (It is same as printf("") in c programming language).



1. **whoami**

Syntax: whoami

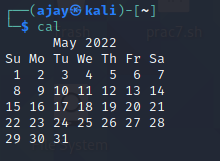
Description: Prints the user name associated with the current effective user Id.

****

1. **cal**

Syntax: cal [options] [month] [year]

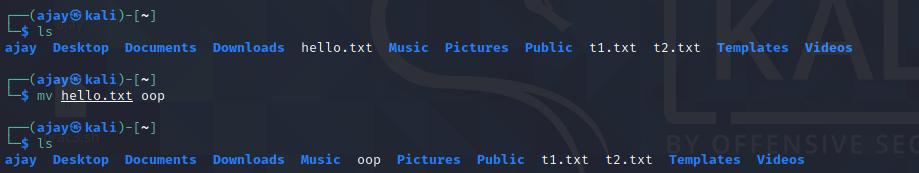
Description: cal displays a simple calendar.



1. **mv**

Syntax: mv [options] old\_name new\_name

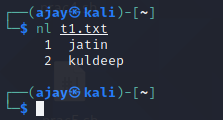
Description: mv is short for move. It has two functions:

* It moves a group of files to a different directory.
* It renames a file.

1. **nl**

Syntax: nl [options] [file\_name]

Description: It numbers the lines in a file.



**Practical-4**

**Aim :- Write a shell script to generate marksheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.**

**Code :-**

echo "Enter marks(out of 30) of "

read -p "Subject 1: " s1

read -p "Subject 2: " s2

read -p "Subject 3: " s3

sum=`expr $s1 + $s2 + $s3`

echo "Sum of marks of 3 subjects is : "$sum per=`expr $sum \\* 10 / 9`

echo "Percentage: "$per if [ $per -ge 60 ]

then

echo "Congratulations, you got Distinction. . ."

elif [ $per -ge 50 ]

then

echo "you got First class. . ."

elif [ $per -ge 40 ]

then

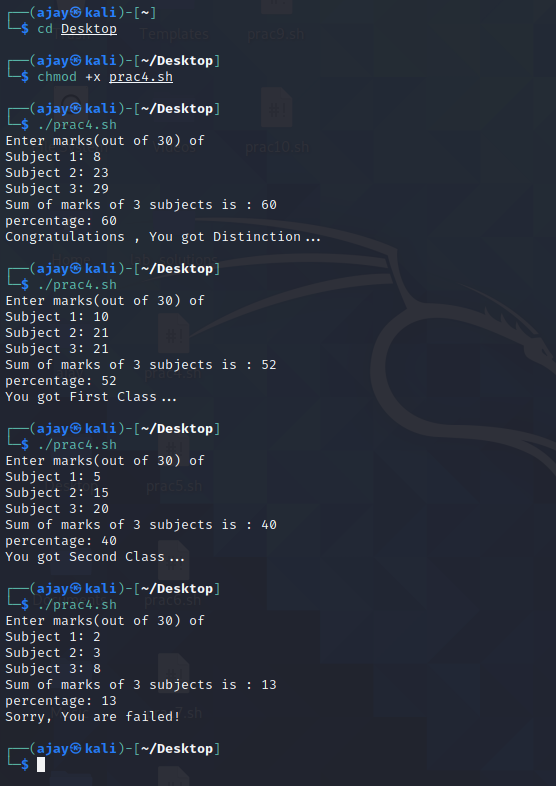
echo "You got second class. . ."

else

echo "Sorry, you are failed! "

fi

**Output :-**

****

**Practical-5**

**Aim :- Write a shell script to display multiplication table of given a number.**

**Code :-**

echo "Multiplication Table "

read -p "Enter the number : " n

i=0

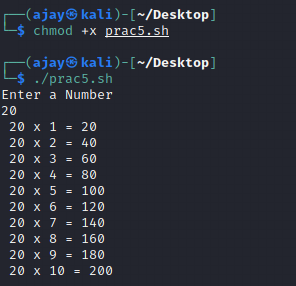
for((i=1;i<11;i++))

do

echo "$n X $i = " "`expr $n \\* $i`"

done

**Output :-**

****

**Practical-6**

**Aim :- Write a shell script to find factorial of given number n.**

**Code :-**

fact=1

read -p "Enter a number to find its factorial : " n

if [ $n -le 0 ]

then

echo "invalid number"

exit

fi

a=$n

while [ $a -ge 1 ]

do

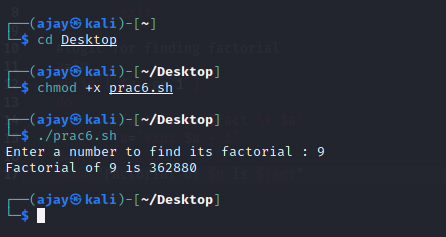
fact=`expr $fact \\* $a`

a=`expr $a - 1`

done

echo "Factorial of $n is $fact"

**Output :-**



**Practical-7**

**Aim :- Write a shell script which will accept a number b and display first n prime numbers as output.**

**Code :-**

read -p "Enter the value of n (or range) " n

echo "The prime numbers are: "

m=2

while [ $m -le $n ]

do

i=2

flag=0

while [ $i -le `expr $m / 2` ]

do

if [ `expr $m % $i` -eq 0 ]

then

flag=1

break

fi

i=`expr $i + 1`

done

if [ $flag -eq 0 ]

then

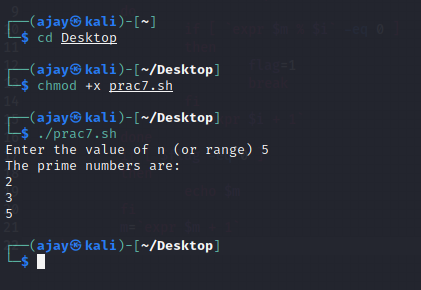
echo $m

fi

m=`expr $m + 1`

done

**Output :-**

****

**Practical-8**

**Aim :- Write a shell script which will generates first n Fibonacci numbers like: 1, 1, 2, 3, 5, 13,…**

**Code :-**

read -p "Enter the value of n (number): " n

x=0

y=1

i=2

echo "Fibonacci Series up to $n terms is : "

echo "$x"

echo "$y"

while [ $i -lt $n ]

do

i=`expr $i + 1`

z=`expr $x + $y`

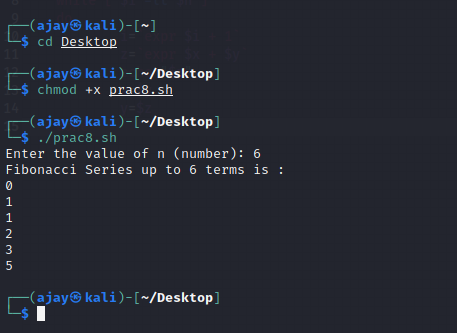
echo "$z"

x=$y

y=$z

done

**Output :-**

****

**Practical-9**

**Aim :- Write a menu driven shell script which will print the following**

**menu and execute the given task.**

**MENU**

**1) Display calendar of current month.**

**2) Display today’s date and time.**

**3) Display usernames those are currently logged in the system.**

**4) Display your name at given x, y position.**

**5) Display your terminal number.**

**Code :-**

i=0

while [ $i != 6 ]

do

echo "Menu

1. Display calender of current Month

2. Display today's date and time

3. Display usernames of those who are currently logged in the ststem

4. Display your name at given x, y position

5. Display Terminal Number

6. Exit

Choose your option and enter corresponding value"

read i

case "$i" in

1. calender="$(cal)"

echo "Here is your Calender "

echo "$calender" ;;

1. current="$(date)"

echo "Current Date and Time is " "$current"

;;

1. username="$(whoami)"

echo "Currently logged in users : "

echo $username

;;

4)

;;

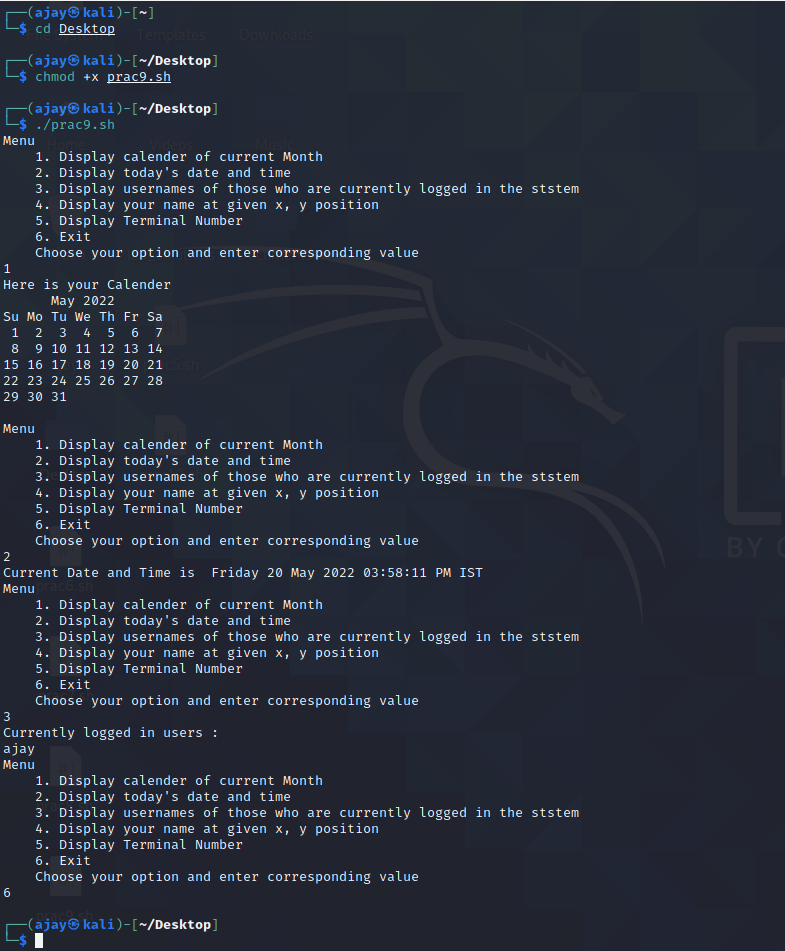
5)

;;

Esac

done

**Output : -**



**Practical-10**

**Aim :- Write a shell script to read n numbers as command arguments and sort them in descending order.**

**Code :-**

read -p "Enter the number of values you want to sort " n

for((i=0; i<$n; i++))

do

read -p "Enter value of arr[$i]: " arr[$i]

done

for((i=0; i<$n; i++))

do

for((j=0; j<n-i-1; j++))

do

if [ ${arr[j]} -lt ${arr[$((j+1))]} ]

then

temp=${arr[j]}

arr[$j]=${arr[$((j+1))]}

arr[$((j+1))]=$temp

fi

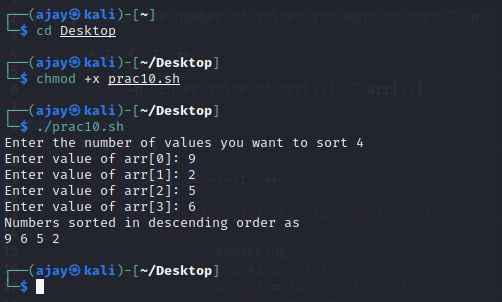
done

done

echo "Numbers sorted in descending order as"

echo ${arr[\*]}

**Output :-**



**Practical-11**

**Aim :- Write a shell script to display all executable files, directories and zero sized files from current directory.**

**Code :-**

echo "Executable files"

files="$(find lab\_solutions -executable -type f)"

echo "$files"

echo

echo "List of Directories"

dir="$(ls -d \*/)"

echo "$dir"

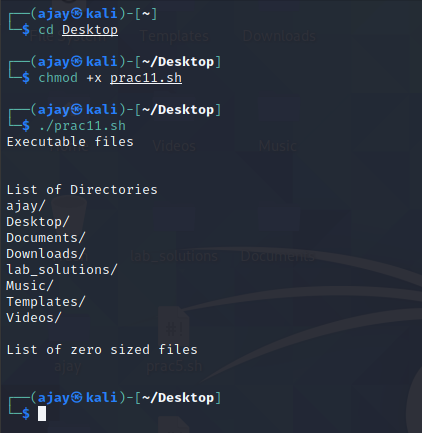
echo

echo "List of zero sized files"

zero="$(find -size 0)"

echo "$zero"

**Output :-**

****

**Practical-12**

**Aim :- Write a shell script to check entered string is palindrome or not.**

**Code :-**

echo "Enter the string to check: "

read str

len=`echo $str | wc -c`

len=`expr $len - 1`

i=1

j=`expr $len / 2`

while test $i -le $j

do

k=`echo $str | cut -c $i`

l=`echo $str | cut -c $len`

if test $k != $l

then

echo "String is not Palindrome"

exit

fi

i=`expr $i + 1`

len=`expr $len - 1`

done

echo "String is Palindrome"

**Output :-**

